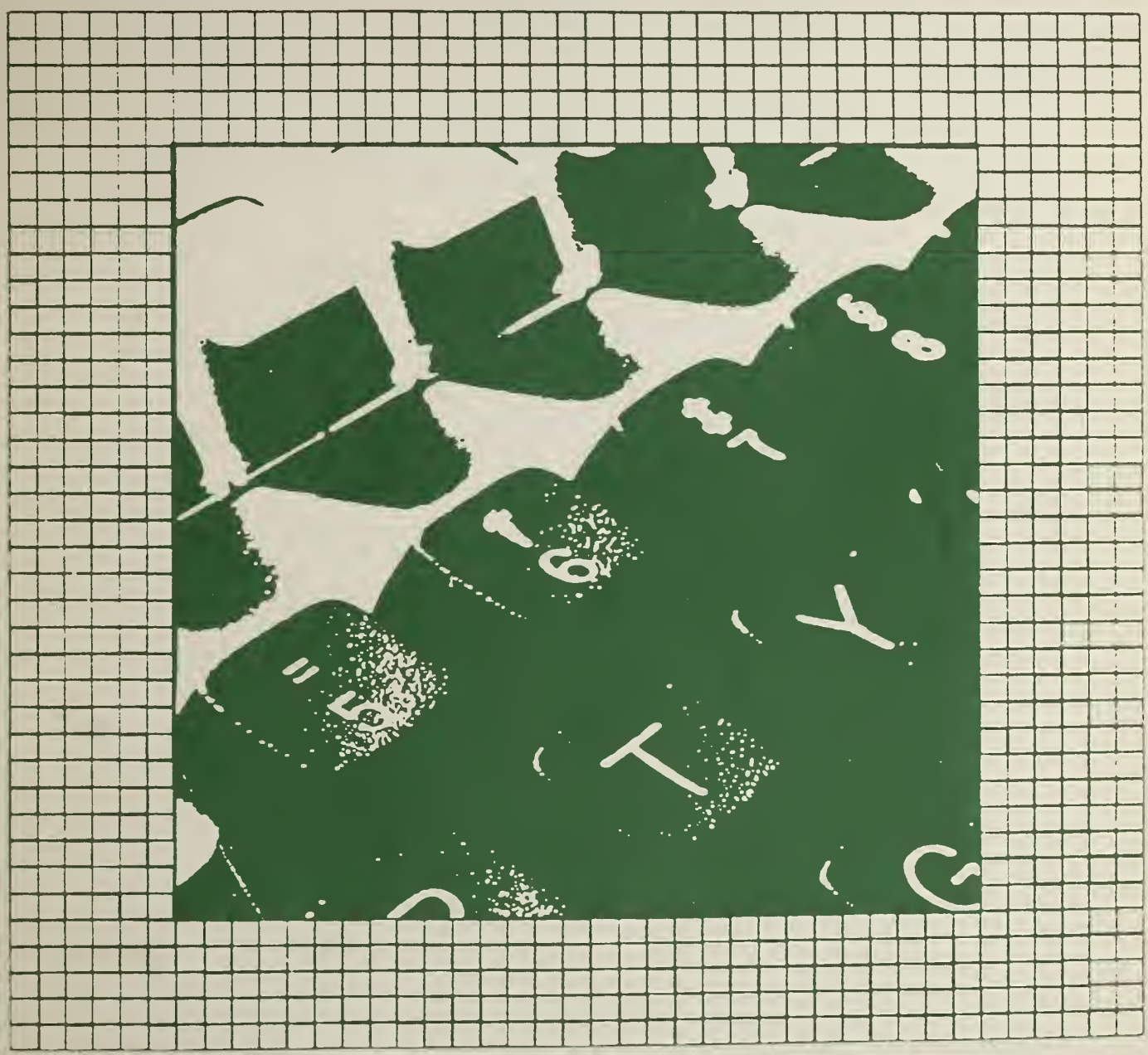
U.S. Department of Transportation

μtps Microcomputers in transportation

Commercial Software Applications For Paratransit



UMTA Technical Assistance Program

NOTICE

DOT-I-84- 51

Commercial Software Applications for Paratransit

Final Report
July 1984

Prepared by
Dynatrend, Incorporated
21 Cabot Road
Woburn, Massachusetts 10801

Prepared for
Office of Methods and Support
Urban Mass Transportation Administration
Washington, D.C. 20590

Distributed in Cooperation with
Technology Sharing Program
Office of the Secretary of Transportation

DOT-I-84-51

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
DOT-I-84-51		MA-06-0039-84-1

4. Title and Subtitle	5. Report Date
Commercial Software Applications for Paratransit	July 1984
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
M. R. Cutler, A. Cabrera, P. A. Monahan, L. J. Harmon	

9. Performing Organization Name and Address	10. Work Unit No.
DYNATREND INCORPORATED 21 Cabot Rd. Woburn, MA 10801	11. Contract or Grant No. DTRS-57-83-C-00109
	13. Type of Report and Period Covered User Manual July 1984
12. Sponsoring Agency Name and Address U.S. Department of Transportation Urban Mass Transportation Administration 400 Seventh Street, S.W. Washington, D.C. 20590	14. Sponsoring Agency Code UMTA

15. Supplementary Notes

*Project Manager. This document is being distributed in cooperation with the Technology Sharing Program of the Office of the Secretary of Transportation.

16. Abstract

This report is a User Manual for the application of commercially available software products to paratransit management. An extensive case study, using the microcomputer database manager, R:base 4000, is documented. It discusses the manual recordkeeping and operational demands placed on paratransit agencies and how these functions can theoretically be automated. It then describes in detail the automation of specific functions including scheduling; client and vehicle recordkeeping; and report generation for billing and performance measurement. Other options are discussed more generally. Data from an actual paratransit operation are used to perform all data processing functions. The focus is on the use of relational database management software. Spreadsheet, word processing and graphics applications are also discussed. Information is provided on how to select software and on how to customize generic software products for specific applications.

17. Key Words	18. Distribution Statement
microcomputers, commercial software packages, record-keeping, billing, scheduling, spreadsheets, word processing, graphics, generic software products	Availability is unlimited. Document is being released to the National Technical Information Service, Springfield, Virginia 22161 for sale to the U.S. public.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
unclassified	unclassified	181	

Form DOT F 1700.7 (8-69)

PREFACE

This report examines the application of commercially available software to paratransit management. An extensive case study, using the microcomputer database manager, R:base 4000, is documented. This project was funded by the U.S. Department of Transportation, Urban Mass Transportation Administration, Office of Methods and Support. The work was performed by DYNATREND Incorporated, under contract to the Transportation Systems Center (TSC).

We would like to acknowledge the support of Mr. Tom Hillegass who directed this project for UMTA, and Mr. Paul Bushueff from TSC. The DYNATREND Project Manager was Mr. Marc Cutler. Technical support was provided by Mr. Antonio Cabrera and Ms. Patricia Monahan of DYNATREND, and by Mr. Lawrence Harman of Call-A-Ride of Barnstable County. The report uses for case study purposes, actual paratransit operating data supplied by Mr. Harman.

Chapter 1.0 describes the manual recordkeeping and operating demands placed on a paratransit agency using Call-A-Ride (CAR) as a model. It also presents a theoretical framework for automating these functions. Chapter 2.0 represents the core of report. It describes in detail the process of using specific software products to automate functions such as scheduling; client and vehicle recordkeeping; and report generation for billing and performance measurement. The focus of this section is on the use of R:base 4000, a relational database management software product. Also discussed more briefly are spreadsheets, word processing and graphics applications. All data processing functions and computer products were generated using the actual data supplied by CAR. Chapter 3.0 discusses, in considerably less detail, other automation options available to users.

This report does not constitute an endorsement of any product or group of products. Given the highly volatile software market, and the range of user applications, users are encouraged to conduct their own in-depth review before selecting a software package for use in their agency. Appendix A provides information to assist users in this process.

Appendices are also provided on the use of floppy disks; customizing the software to better perform specific functions; screen forms to assist users in replicating the functions described in the manual; blank forms for future use; and an update on recent changes in R:base 4000.

Note: In a separate but coordinated project, a similar client file, scheduling and reporting system has been developed using the popular DBASE II microcomputer database management program. All the user programs and documentation for this menu-driven system are available from the Government for a nominal copying fee. (This does not include the DBASE II program itself, which is a proprietary product and must be purchased elsewhere for about $500.) This system is offered as a fully operable example, to be customized by the paratransit operator with the help of a local consultant or other support person skilled in DBASE II implementation. Contact the Transportation Systems Center at (800)225-1612 and ask about "SST" for information.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

APPENDICES

LIST OF FIGURES

LIST OF FIGURES (Concluded)

LIST OF TABLES

1.0 PARATRANSIT MANAGEMENT: MANUAL RECORDKEEPING AND THE POTENTIAL FOR AUTOMATION

This section describes the manual recordkeeping demands placed on a paratransit agency and presents a theoretical model for automating these functions. The description of the manual recordkeeping system is based on an actual paratransit operator, Call-A-Ride of Barnstable County (CAR). CAR serves as a case study throughout the remainder of this Manual.

1.1 CALL-A-RIDE'S MANUAL RECORDKEEPING AND REPORTING REQUIREMENTS

1.1.1 Background

Between 1975 and 1979, CAR, a private non-profit corporation, provided consolidated human services transportation for residents of Cape Cod. General public transportation service was initiated in 1978.

CAR's service area included fifteen towns and covered approximately 394 square miles. For the first four months of 1978, twenty-five vehicles were in operation. In May of that year, the number of vehicles in-service was reduced to fourteen. Approximately 10,000 trips per month were provided to destinations such as health care facilities, congregate meal sites, adult day care centers, and special education facilities. In addition, deliveries of food for a "meals-on-wheels" program were made by CAR.

CAR's annual operating costs for this consolidated human services transportation program averaged about $213,000. Funding was obtained from a variety of federal, state, and local sources, including: Section 16(b)(2); Titles III, VII (Older Americans Act), XIX, XX (Social Security Act); Massachusetts Chapter 766; and CETA.

All information pertaining to CAR's daily operations (vehicle schedules, maintenance and repair records, and operations reports; client trip requests) and more permanent data (client record file) was compiled, recorded, and updated manually, as is the case in most paratransit organizations. Invoices, requests for funding, and periodic statistical reports were also prepared manually by extracting information from paper files. The following subsections provide an overview of these recording and reporting functions in order to identify the specific types of information that CAR and other paratransit agencies need to record and store, and the uses to which that information is put in the management of paratransit operations.

1.1.2 Client Records

The client record file contains a permanent record of information about each client or passenger who uses or has used a paratransit agency's services. Included is information such as: name, address, phone number; birthdate; passenger classification; Medicaid, Title XX, or other funding source authorization or number; and any special needs or other comments.

In a manual system of recordkeeping, the client record file is usually stored on cards similar to the one shown in Figure 1-1, and contains a record of each trip taken by a client as well as his/her personal data. Records for new

E EH H NU PHONE # _____ MALE/FEMALE

CASE CLOSED: REASON _____
DATE _____

NAME _____

ADDRESS _____

_____

MEDICAID # ___-_____-___-_____-___-_____

TITLE XX # _____ DATE EXPIRED _____

DATE	#	TRIP		DATE	#	TRIP		DATE	#	TRIP		DATE	#	TRIP	

FIGURE 1-1: SAMPLE CLIENT RECORD

clients are added to the file at the time of an initial trip request. In the CAR system, all client records were updated at the conclusion of each day of operations, so that current information on trips taken by each client was available at all times.

The principal reasons for recording client trip and personal data are 1) to form the basis for client-specific billing; 2) to supply client-specific documentation for audit purposes; 3) to provide a monthly count of unduplicated passengers for Section 16(b)(2) reports; and 4) to aid in the preparation of driver itineraries.

1.1.3 Vehicle Scheduling

The process used for the manual scheduling of daily paratransit vehicle trips is depicted in Figures 1-2, 1-3, and 1-4. In the CAR model, the scheduling process follows one of two paths, depending on whether the trip is pre-scheduled (a standing appointment usually between a client's home and a human services agency activity center) or a dial-a-ride request.

In the pre-scheduled mode (Figure 1-2), a trip request is received through a program or agency or directly from a client. The dispatcher then follows the steps listed below:

- Identify client and verify eligibility
- Determine trip date parameters: for example, trips for medical purposes such as physical therapy or chemotherapy/radiation treatment typically have specified start and end dates, and may need to meet strict time requirements for arriving at the facility.
- If the client is new to the system, enter client data into the client record file; i.e, fill out a client record card.
- Go to schedule process.

In the demand-responsive mode (Figure 1-3), an individual phones the paratransit agency with a request for a one-time trip, and the preliminary scheduling procedure is as follows:

- Identify client and verify eligibility
- Determine trip purpose: does the request fit the agency's priority trip requirements (e.g., medical)?
- If client is a new user, fill out a client record card.
- Go to schedule process.

At this point in the construction of a vehicle schedule, the steps followed in the pre-scheduled and dial-a-ride modes are identical (Figure 1-4):

- Determine passenger classification, e.g., elderly, elderly/handicapped, non-elderly/handicapped, non-elderly/non-handicapped.
- Enter day of trip on dispatcher schedule or trip request form.
- Enter inbound origin, destination, and appointment time, if any.
- Determine approximate inbound pick-up time.
- Note any special equipment or assistance needed: does the client use a wheelchair? Does he/she need assistance negotiating steps?

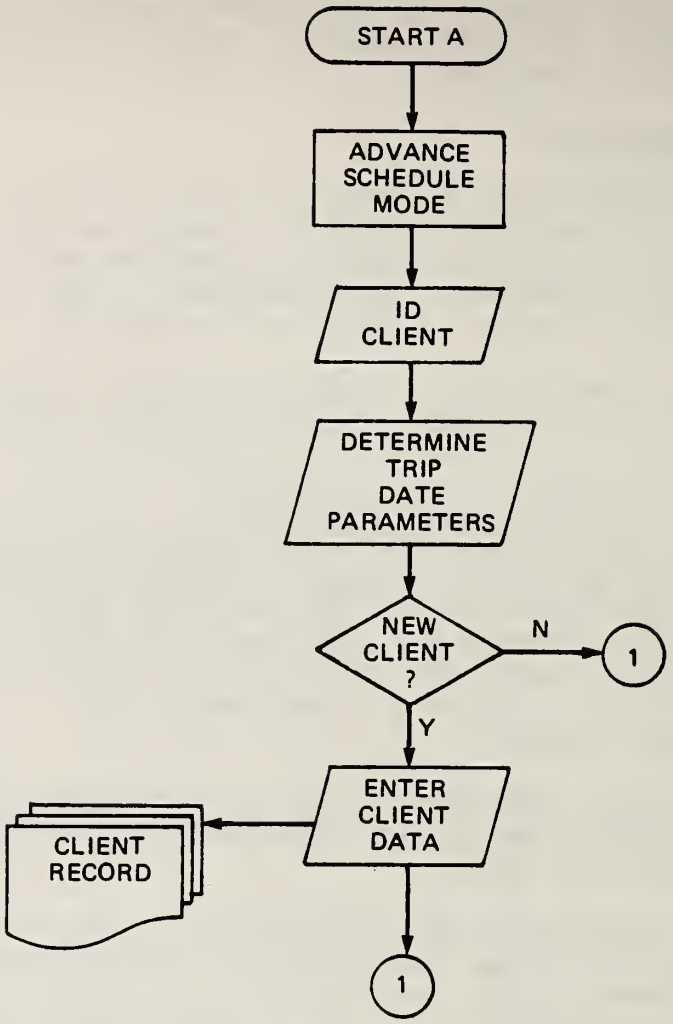


FIGURE 1-2: PRELIMINARY SCHEDULING PROCEDURES, PRE-SCHEDULED TRIPS

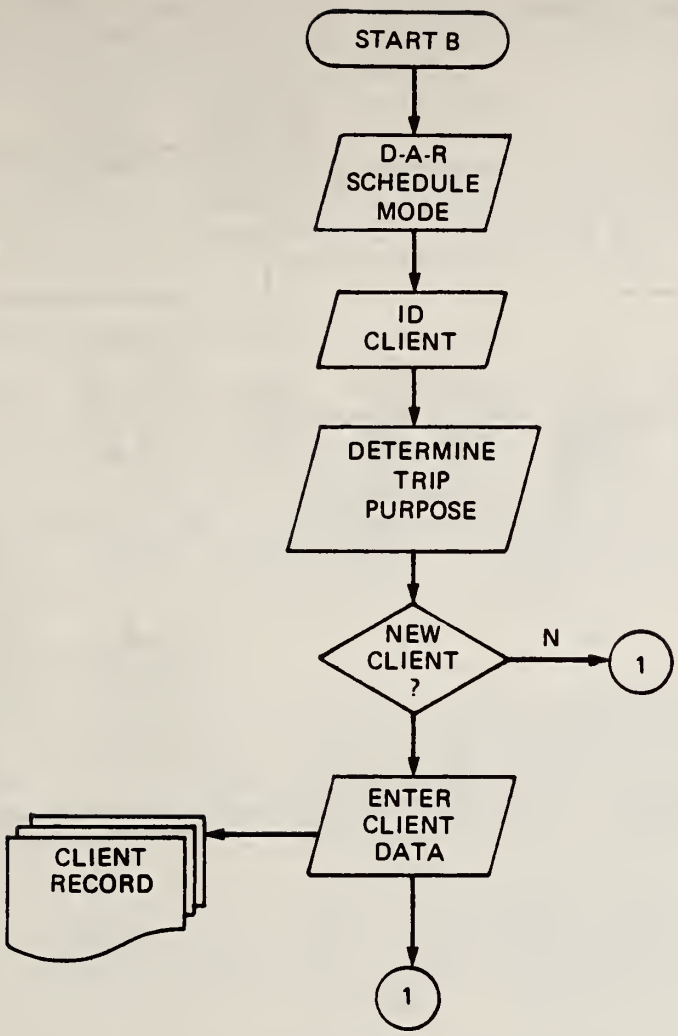


FIGURE 1-3: PRELIMINARY SCHEDULING PROCEDURES, DIAL-A-RIDE TRIPS

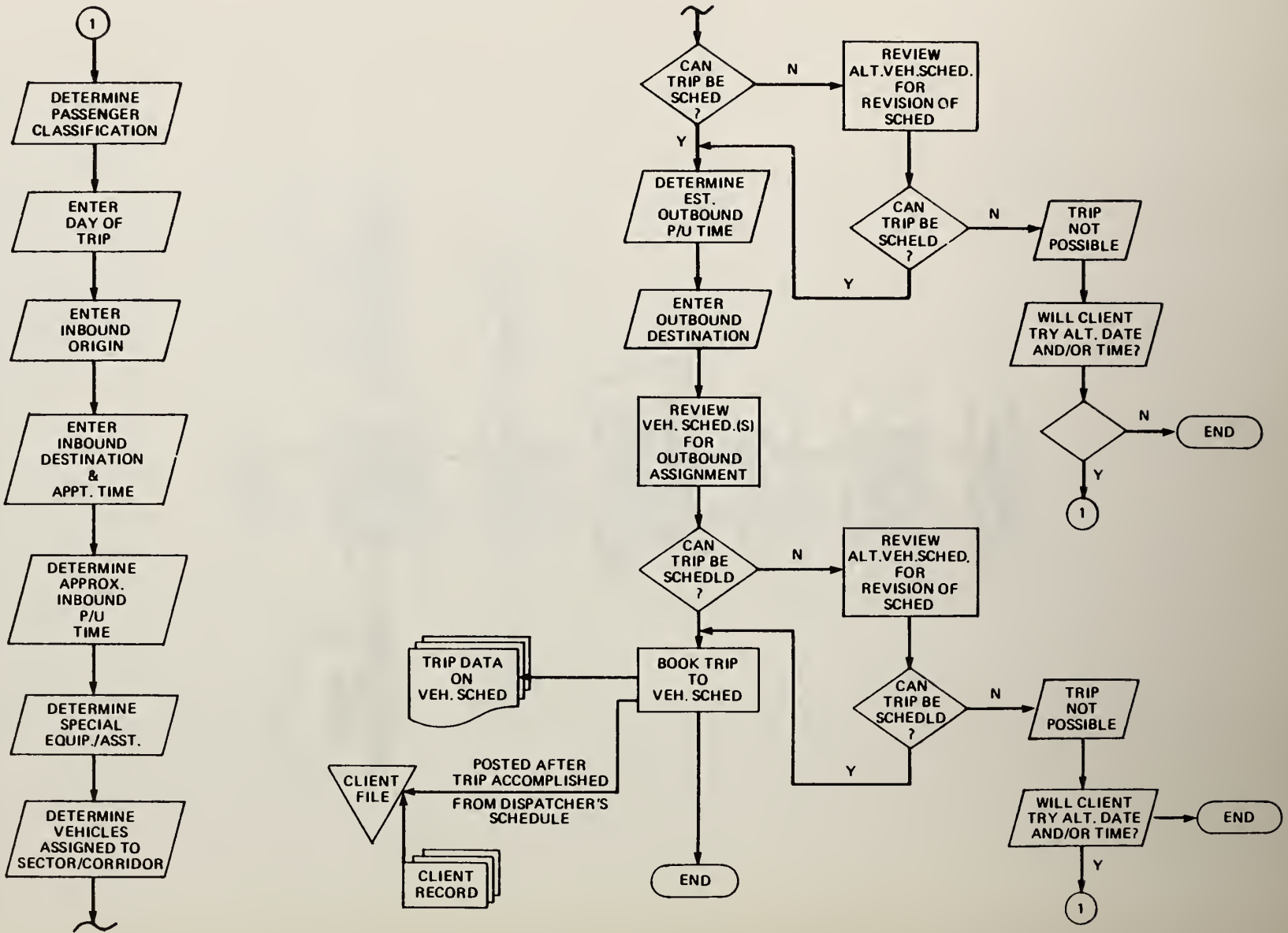


FIGURE 1-4: MANUAL PARATRANSIT SCHEDULING PROCESS

- Review schedules of vehicles assigned to appropriate zone for inbound assignment.
- If possible, schedule trip; if not, rearrange vehicle schedules to accommodate trip or arrange alternate day and/or time with client.
- Determine outbound destination and approximate pick-up time*.
- Review schedules of vehicles assigned to appropriate zone for outbound assignment.
- If possible, schedule trip; if not, rearrange vehicle schedules to accommodate trip or arrange alternate day and/or time with client.
- Enter trip data on vehicle schedule.
- Record trip data on client record card.

A sample vehicle schedule format is displayed in Figure 1-5. In the CAR system, all vehicle schedules are photocopied at the end of the day so that one copy can be given to the driver of each vehicle as an itinerary while the original record remains on file at the dispatch center.

1.1.4 Vehicle Operating Statistics

In the CAR model, daily and weekly vehicle condition and mileage reports (see Figure 1-6) are completed by each driver. These reports are used to collect data pertaining to vehicle miles by contract or program (as required for billing purposes), and the cost and quantity of gas and oil used. An attempt was made on the part of CAR's management to minimize data collection by vehicle operators, thereby allowing them to concentrate on providing safe and effective transportation service for CAR's clients.

1.1.5 Client/Agency Billing

As do most paratransit agencies incurring costs of providing service that are reimbursable on a client-specific basis, CAR tracked individual client trips so as to be able to bill human service organizations accurately for the CAR services utilized by each organization's clients. Therefore, monthly or quarterly invoices were derived manually from the client record file.

Figure 1-7 is an example of an invoice for special education program trips prepared using data contained in the client record file - in this case, passenger-miles traveled based on the number of trips and the accumulated mileage between the origins and destinations listed on each client's record.

On the basis of past service utilization, expected passenger-miles for the upcoming quarter were calculated for each client whose trips were paid for by the Cape Cod Collaborative. The town in which each client's trip originated was noted. Passenger-miles for each town were summed. Finally, the costs of providing this transportation service were allocated among the member towns in proportion to each town's share of the total passenger-miles. For instance,

*Given the rural nature of the Cape Cod service area and CAR's operating philosophy (guarantee every return trip, make all operating decisions as early as possible, but remain flexible), the agency's practice was to book each return trip at the time of the initial trip request.

CALL-A-RIDE OF BARN[illegible] BLE COUNTY, INC. DIS[illegible] CHER'S SCHEDULE

Time Codes: X = Drop-Off
P = Pick-Up

Person Codes: E = Elderly(60+ years)
EH = Elderly-Handicapped
H = Non-Elderly-Handicapped

Program Codes: (Trip Purpose)
HC = Health Care
NU = Site Nutrition
MOW = Meals-On-Wheels
SE = Special Education
FT = Special Education Field Trips
OT = Group Trip for the Elderly
ADC = Adult Day Care
DL = Dialysis

TIME	CODE	NAME AND ADDRESS	PERSON CODE	PHONE	TRIP PURPOSE	DESTINATION	COMMENTS
1430	P						
1430	P						
1430	P						
1430	P						
1430	P						
"	P						
"	P						
"	P						
"	P						
"	P						
"	P						
"	P						
"	P						
"	P						
"	P						

DAY WEDNESDAY DATE JAN 3 1979

FIGURE 1-5: SAMPLE DAILY VEHICLE SCHEDULE

932 mi.
$79.50 gas
80.0 gal.

CALL-A-RIDE OF BARNSTABLE COUNTY, INC.

DRIVER'S DAILY & WEEKLY VEHICLE CONDITION & MILEAGE REPORT

Date: Wk. Ending 3-23 Mobil No. 69

Gasoline & Cost Per Gals.

Mon. 3-19	Tues. 3-20	Wed. 3-21	Thur. 3-22	Fri. 3-23
A. Cost 13.00	A. Cost 13.50	A. Cost 15.00	A. Cost 10.00	A. Cost 15.00
B. Gals. 17.6	B. Gals. 18.3	B. Gals. 20.3	B. Gals. 13.5	B. Gals. 20.3

Location Falmouth

Oil - Cost Per Quart

Mon.	Tues.	Wed.	Thur.	Fri.
A. Cost	A. Cost	A. Cost	A. Cost	A. Cost
B. Qts.	B. Qts.	B. Qts.	B. Qts.	B. Qts.

		Program: Specialized		Program: Nutrition		Program: Craigville Kids	
		Tour 1	Tour 2	Tour 1	Tour 2	Tour 1	Tour 2
Monday	Start	66439	66575	66508			
	Finish	66503	66606	66575			
Tuesday	Start	66606	66751	66685	66728	66714	
	Finish	66685	66800	66714	66751	66728	
Wednesday	Start	66800	66938	66890			
	Finish	66890	66963	66938			
Thursday	Start	66963	67147	66203			
	Finish	66703	67179	67149			
Friday	Start	67179	67341	67264			
	Finish	67264	67371	67341			
Saturday	Start						
	Finish						
Sunday	Start						
	Finish						

Special ED: 55
Nut: 130
Craigville: 14

FIGURE 1-6: SAMPLE VEHICLE OPERATIONS REPORT

EXHIBIT III-A-4

Cape Cod Regional Transit Authority, 215 Iyanough Road, Hyannis, MA 02601 (617)775-2696

May 31, 1978

INVOICE

For transportation services to be provided to the Cape Cod Collaborative from June 1, 1978 through August 31, 1978:

TOWN	COST
Wareham	$1,369.44
Bourne	$1,069.76
Mashpee	$557.92
Barnstable	$3,395.56
Yarmouth	$3,550.40
Dennis	$1,862.96
Chatham	$912.96
TOTAL	$12,680.00

Lawrence J. Harman

Lawrence J. Harman, General Manager

FIGURE 1-7: SAMPLE AGENCY INVOICE

EXHIBIT III-A-4
(CONTINUED)

Anticipated Passengers and Passenger-Miles for Fourth Quarter, 1978

Name	Program	Expected 4th Qtr. Days	2-way Miles per Day	Total 4th Quarter Miles	Town
Joe R.	[illegible]/CE	12	51	612	Wareham
David C.	[illegible]/H	65	87	5655	Wareham
Sly D.	MS/CE	12	18	216	Bourne
Linda B.	[illegible]/H	65	70	4550	Bourne
Shontell S	NH	11	27	297	Mashpee
Betty B.	[illegible]/H	65	35	2275	Mashpee
Barbara C.	PV	65	57	3705	Barnstable
Roger B.	PV	65	55	3575	Barnstable
Calvin M.	PV	65	43	2795	Barnstable
Barbara S.	PV	65	43	2795	Barnstable
Joyce H.	PV	65	39	2535	Barnstable
Linda R.	PV	65	69	4485	Yarmouth
Harry S.	PV	65	71	4615	Yarmouth
Scott G.	[illegible]/H	65	22	1430	Yarmouth
Mary O.	[illegible]/H	65	20	1300	Yarmouth
Scott S.	[illegible]/H	65	14	910	Yarmouth
Alex F.	[illegible]/H	65	16.4	1066	Yarmouth
Nancy G.	[illegible]/H	65	8.4	546	Yarmouth
Brian K.	[illegible]P/H	12	24	288	Yarmouth
Barbara D.	[illegible]P/H	12	24	288	Yarmouth
Diane L.	[illegible]P/H	12	24	288	Yarmouth
Terry O.	[illegible]/H	65	14.4	936	Yarmouth
Joyce K.	[illegible]/O	65	43.4	2821	Dennis
Mary C.	[illegible]/H	65	38	2470	Dennis
Carrol R.	[illegible]/H	65	28	1820	Dennis
Phillip J.	[illegible]/H	65	20.8	1352	Dennis
Frank D.	[illegible]/O	65	27.8	1807	Chatham
Ann D.	[illegible]/O	65	20.2	1313	Chatham
Jerry W.	[illegible]/O	65	16	1040	Chatham

FIGURE 1-7: SAMPLE AGENCY INVOICE (cont'd)

EXHIBIT III-A-4
(CONCLUDED)

Cape Cod Collaborative

Cost Allocation - 4th Quarter

Town	Passenger-Miles 4th Qtr.	Proportion of Total Miles	Cost
Wareham	6,267.0	0.108	$1,369.44
Bourne	4,766.0	0.082	1,039.76
Mashpee	2,572.0	0.044	557.92
Barnstable	15,405.0	0.267	3,385.56
Yarmouth	16,152.0	0.280	3,550.40
Dennis	8,463.0	0.147	1,863.96
Chatham	4,160.0	0.072	912.96
TOTALS	57,785.0	1.000	$12,680.00

- Six passengers will be transported for only the first two weeks of the fourth quarter due to the completion of the school year. The program cost for this summer schedule is estimated to be $12,680.00 or 22¢ per passenger-mile. Total program cost for the FY78 program is $65,824.58.

- The increased proportion of cost for the Town of Yarmouth reflects the relocation of a former Wareham passenger and the addition of one passenger.

FIGURE 1-7: SAMPLE AGENCY INVOICE (concl'd)

Wareham, with 10.8% of the total passenger-miles, was assessed 10.8% of the total cost of providing this transportation service, or $1,369.

This invoice is but one example of the various billing formats and procedures specified contractually by the administrators of CAR's different funding sources. Transportation services provided under some programs were reimbursable at a negotiated rate per vehicle hour (nutrition program) or per unit cost of authorized service (Medicaid Title XX trips). Other funding sources provided operating assistance to CAR in grant form (Title III).

Complicating the billing process further were conflicts between the client eligibility and authorization policies of various funding sources and the differences in the billing cycles of the agencies involved. Given all these billing complexities, maintenance of an up-to-date client record file containing accurate information on each client and the details of his/her utilization of CAR's services was absolutely essential to the smooth administrative and financial functioning of the organization.

1.1.6 Budget Preparation

It is important to note at the outset that the brief review of CAR's budgeting process presented in this subsection focuses on the types of data required to produce a paratransit agency budget, rather than CAR's particular budgeting techniques or philosophy.

The preparation of CAR's annual combined program operating budget began with a mapping of each vehicle's expected schedule by time of day and program, as illustrated below in Figure 1-8.

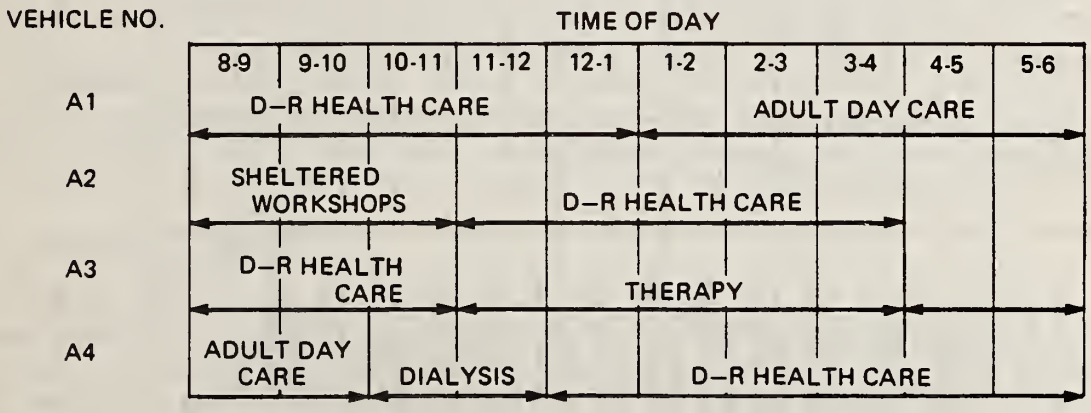


FIGURE 1-8. HOURLY VEHICLE UTILIZATION CHART BY PROGRAM

This chart was then collapsed into a daily/weekly utilization chart, such as the one shown in Figure 1-9. The vehicle utilization chart represented an analysis of the characteristics of present and prospective programs to determine vehicle availability, the spatial relationships between clients and service facilities, and required service levels (in terms of person trips) for each. In addition to serving as a planning tool for maximizing vehicle

VEHICLE UTILIZATION CHART

EXHIBIT II-E-3
(Continued)

Vehicle Code	Vehicle Description	D-R Health Care	Therapy	Adult Day Care	Dialysis	Site Nutrition	HDM Delivery	Sheltered Workshops	Off-Cape Medical (Boston)	Off-Cape Medical (Pondville)	Total Vehicle Hours
A1	1976 Dodge Maxivan (lift-equipped)	5		5							10 (4 days/wk.
A2	1976 Dodge Maxivan (ramp-equipped)	5						3			8
A3	1976 Dodge Maxivan (lift-equipped)	3	5	2							10 (4 days/wk.
A4	1976 Dodge Maxivan (ramp-equipped)	5.6		2	2.4						10 (4 days/wk.
A5	1976 Dodge Maxivan (ramp-equipped)	X X X	X X	X X X	X X	SPARE BUS		S X X	X X X	X X	X X X X X X
B1	1979 Dodge Maxivan (lift-equipped)			3		4	1				8
B2	1979 Volkswagen Bus (7-passenger)					3.5	1.5				5
B3	1977 Dodge Maxivan (12-passenger)					4	.5	5.5			10 (4 days/wk.)
B4	1977 Dodge Maxivan (15-passenger)					4.5	.5	2			7
B5	1977 Dodge Maxivan (12-passenger)					6					6
C1	1976 Dodge Maxivan (12-passenger)						7				7
C2	1979 Volkswagen Bus (7-passenger)									6	6
C3	1977 Dodge Maxivan (lift-equipped)								4		4
C4	1979 Volkswagen Bus (7-passenger)	X X X	X X X	X X	X X X	SPARE BUS		S X X	X X X	X X	X X X X X X
	TOTALS	18.6	5	12	2.4	22	10.5	10.5	4	6	91

CALL-A-RIDE OF BARNSTABLE COUNTY, INC.

FY 79 WORK PROGRAM

FIGURE 1-9: SAMPLE WEEKLY VEHICLE UTILIZATION CHART BY PROGRAM

utilization and allocating driver time, this chart provided a mechanism through which to allocate indirect costs and multi-program direct costs to funding sources in the budgeting process.

Based on the vehicle utilization chart, direct costs by program (e.g., health care, nutrition) were estimated. Indirect costs were then estimated and allocated among programs according to each program's share of total direct costs. Finally, estimated income by program was determined.

In order to construct such a budget, CAR utilized extensive data pertaining to vehicle utilization, service utilization, income, and expenditures by program.

1.1.7 Reporting Requirements and Program Evaluation

As an operator of vehicles purchased with Section 16(b)(2) funds, CAR was required to complete and submit the statistical report displayed in Figure 1-10 to the Massachusetts Executive Office of Transportation and Construction (EOTC) every month. This report is a typical example of the data collection and reporting requirements placed on paratransit agencies by funding sources for monitoring and evaluation purposes.

1.2 MICROCOMPUTER APPLICATIONS FOR PARATRANSIT

As evidenced in the preceding sections, information management is an essential function in the delivery of paratransit service. Complete and accurate records must be maintained on passengers, trips, and vehicles in order to: 1) provide service to clients, and 2) facilitate the agency's accounting and financial management activities.

Because of the amount of information to be maintained in the data base of a typical paratransit organization such as CAR, and because of the inter-relationships that exist between data files, one of the most useful pieces of microcomputer software for an organization to acquire is a relational data base management system (DBMS). Other valuable computing tools are spread-sheet, word processing and graphics packages. The organization of these four software packages into a system of microcomputer-based paratransit management applications is discussed in more detail in the following subsections.

1.2.1 Conceptual Design

A relational DBMS, as opposed to a file management system, treats data as if it were stored in tabular form, recognizing the relationships between data items and data sets. Information is contained in fields (such as a client's name); records consist of fields that are related to each other (a client's name, address, phone number, passenger classification, ID number, and so forth). Files are made up of records that share the same field structure (i.e., a paratransit agency's master client file). A collection of related files is a data base (i.e., master client file, vehicle schedule file, vehicle operations file, etc.).

In any data base, different files may contain common fields, or fields that are dependent on each other for definition. A relational DBMS is capable of manipulating fields and files in two unique ways, due to its tabular

EXHIBIT III-B-1

Call-A-Ride of Barnstable County, Inc.

MONTHLY STATISTICAL REPORT

Health Care and Nutrition Transportation Services

Prepared by Russell H. Chatfield Month, Year June, 1978

I. DIRECT SERVICE STATISTICS BY FUNDING SOURCE

A. Health Care Transportation

	T.	III	XIX	XX	TOTAL
1. # UNDUPLICATED passengers this month		158	71	26	255
2. # New passengers this month		38	15	1	54
3. # Year-to-date UNDUPLICATED passengers		575	132	27	734
4. # UNDUPLICATED elderly this month		95	6	19	120
5. # UNDUPLICATED elderly-handicapped this month		57	50	6	113
6. # UNDUPLICATED non-elderly handicapped this month		7	15	0	22

B. Nutrition Transportation

1. # UNDUPLICATED passengers this month	214
2. # New passengers this month	21
3. # Year-to-date UNDUPLICATED passengers	370
4. # UNDUPLICATED elderly this month	146
5. # UNDUPLICATED elderly-handicapped this month	68

II. FREQUENCY OF SERVICE BY FUNDING SOURCE

A. Health Care Transportation

	T.	III	XIX	XX	TOTAL
1. # Of one-way medical trips this month		736	560	125	1421
2. # Of elderly medical trips this month		333	49	70	452
3. # Of elderly-handicapped medical trips this month		305	324	55	684
4. # Of non-elderly handicapped med. trips this month		98	187	0	285
5. # Of no-shows this month					5
6. # Of not possible trips this month					10
7. # Of missed trips this month					0

No. of Passengers	No. of one-way trips
194	0-4
24	5-10
8	11-15
12	16-20
17	21 +

TRIP ORIGINS

Barnstable	491	Mashpee	65
Bourne	202	Orleans	4
Brewster	14	P'town	0
Chatham	0	Sandwich	15
Dennis	105	Truro	0
Eastham	0	Wellfleet	0
Falmouth	295	Yarmouth	191
Harwich	39		

B. Nutrition Transportation

1. # Of congregate nutrition trips this month	2960	
2. # Of shopping assistance trips this month	132	
3. # Of home delivery trips this month	424	
4. # Of other trips this month	18	3534
5. # Of no-shows this month	5	

III. VEHICLE STATISTICS

	Health	NUTRITION Sites	NUTRITION Food Del.
1. Total miles driven	11,321.2	6,984.1	4,872.1
2. Average miles driven per vehicle per day	128.65	59.86	44.24
3. Total vehicle-hours	604.0	467.3	295.7

FIGURE 1-10: SAMPLE STATISTICAL REPORT

organization of data. First, separate files containing common fields can be merged, allowing access to a great deal of related data at one time. For instance, a daily vehicle operations file and a vehicle maintenance and repair file, each including a field for vehicle ID number, could be joined in order to determine operating cost by vehicle. Or, a vehicle schedule file containing a client ID number field could be merged with the master client file, also containing client ID number, so that detailed information on an individual client and his/her trips could be used for billing purposes.

Second, changing a field in one file in a relational DBMS automatically causes all identical fields in other files to be changed accordingly. Thus, changing or updating a field that appears in several files, such as client ID number, can be accomplished with a single command. For more detailed information on the different types of data base management software, see Appendix A, "Selecting a Data Base Manager".

Automation of the paratransit agency management functions described in Section 2.0 should be centered around relational DBMS files and applications, but would also involve the development of spreadsheet, word processing, and graphics.

The overall organization of a microcomputer-based automated system is illustrated in Figure 1-11, which indicates the primary files to be developed for use by each software component. The ideal automated system would use an integrated software package whose relational DBMS, spreadsheet, word processing, and graphics programs each generate files that can be read and utilized by the other programs with the least amount of user intervention. (See Appendix A for the pros and cons of currently available levels of integration in microcomputer software.)

1.2.2 Relational DBMS Files and Applications

The files to be created and maintained in the relational DBMS are shown in Figure 1-11. They include:

- Master Client File: a list of all present and past paratransit service users, including such pertinent data as address, phone number, age, passenger classification, Medicaid number, human services agency authorization, and special instructions.

- Master Vehicle File: a record of all vehicles purchased, including license number; serial number; year, make, and model; engine size; owner; conversion information; and purchase date.

- Pre-scheduled Trips File: a "dummy" file for a day(s) of pre-scheduled trips for each vehicle, which can be edited (to alter individual vehicle assignments and service date) and merged with the active data base to be used in the preparation of daily vehicle schedules.

- Daily (real time) Vehicle Schedule File: a record of daily vehicle trips, including vehicle and client ID numbers, scheduled pick-up and drop-off times, origin and destination addresses, passenger classification, trip purpose, and special needs.

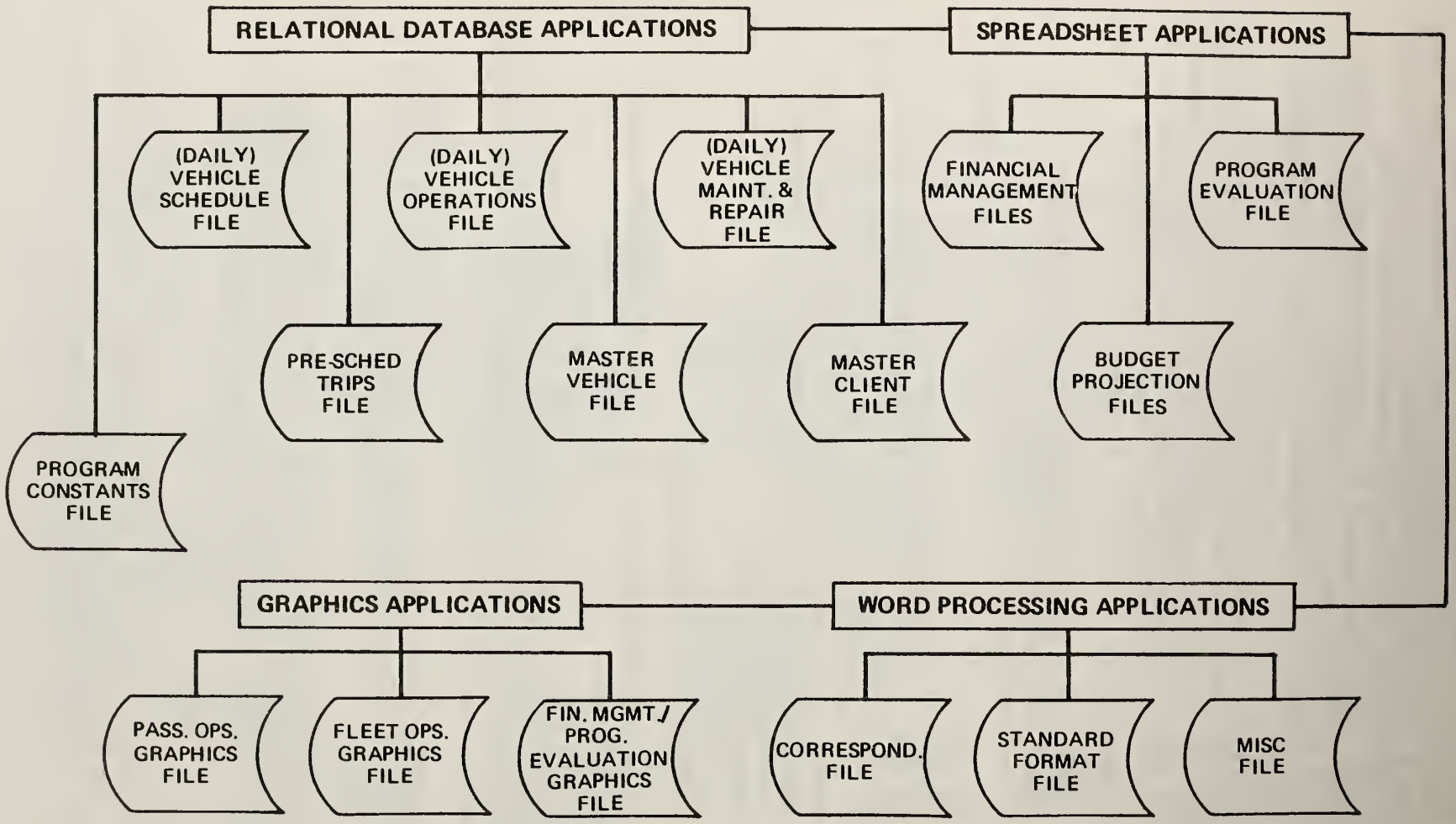


FIGURE 1-11: CONCEPTUAL DESIGN OF MICROCOMPUTER APPLICATIONS

- Daily Vehicle Operations File: daily operating statistics for each vehicle in-service, including driver and vehicle ID numbers; starting and ending mileage; start time and end time; quantity and cost of fuel, oil, coolant, and transmission fluid used; necessary repairs or road calls; and contract vehicle hours. The vehicle operations report is filled out by each driver during his/her daily run and entered into the data base later.

- Daily Vehicle Maintenance and Repair File: a record of maintenance and repairs performed on each vehicle, including a description of work done, cost of parts and labor, and vendor.

These files would be used alone and in combination to produce driver itineraries, client/agency invoices, and management tools such as operations reports, vehicle maintenance reports, and ridership reports, as described below.

1.2.2.1 Vehicle Scheduling

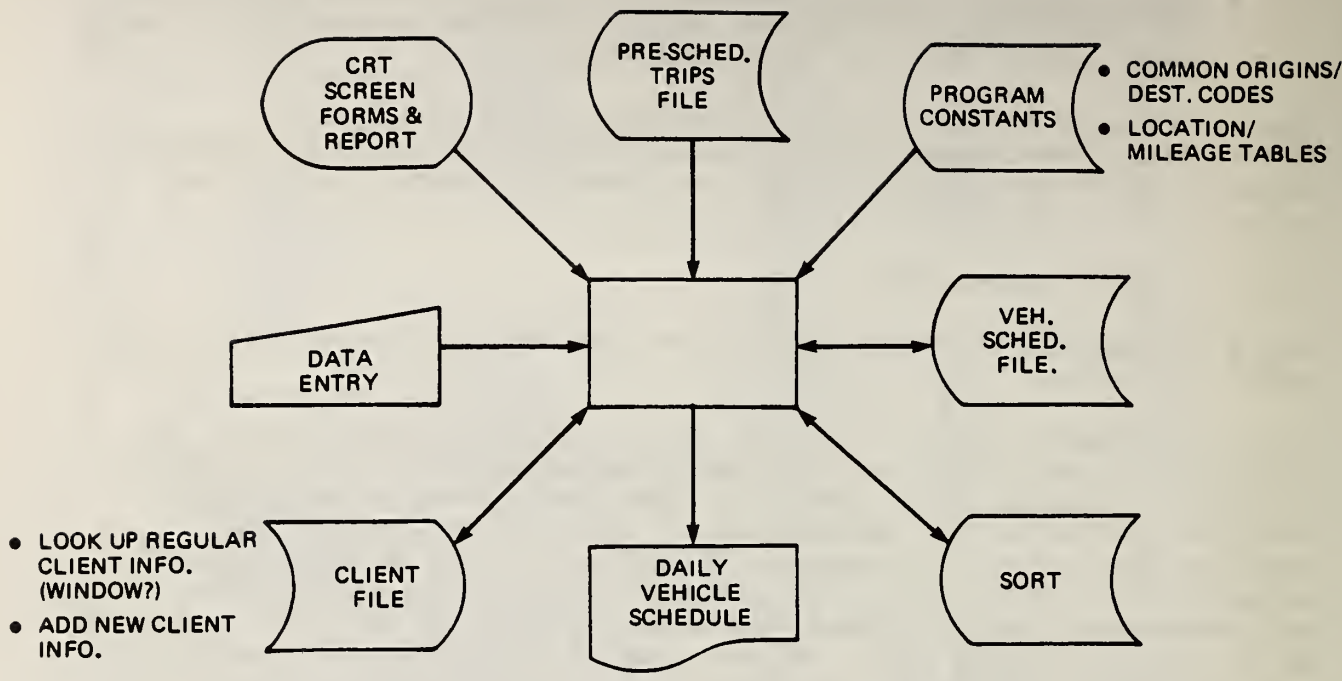
Figure 1-12 depicts the various DBMS files that would be used to create daily vehicle schedules.

First, the scheduler creates or edits the pre-scheduled trips file for the particular day/month/year, and adds the pre-scheduled trips for that date to the vehicle schedule file. Then, in an iterative process at appropriate intervals, he/she reviews the schedules of suitable vehicles and adds real-time dial-a-ride trip requests for the service date to the vehicle schedule file. Information on new clients is added to the master client file, and information from the updated file on both current and new clients is added to the vehicle schedule file. The next step is to prepare daily individual vehicle schedule reports for the service date from the vehicle schedule file, a copy of which is given to the driver to serve as his/her itinerary. At the conclusion of the day's operations, the scheduler updates, or edits, the vehicle schedule file for that day.

It is important to note that the assignment of trips to each vehicle is still done by the scheduler. In this system, the microcomputer software is used only to handle the information necessary for vehicle scheduling quickly and efficiently, thereby making the scheduler's job easier.

1.2.2.2 Billing

Figure 1-13 illustrates the interaction of DBMS files to produce client or agency invoices. Using the data listed below from each file, the paratransit agency's financial management staff would be able to generate standard invoices for prescribed periods (e.g. monthly) for program activity in accordance with contractual billing rates.



SCHEDULING PROCEDURE:

① Scheduler creates/edits a PRE-SCHED. TRIPS FILE for date: yr/mo/day

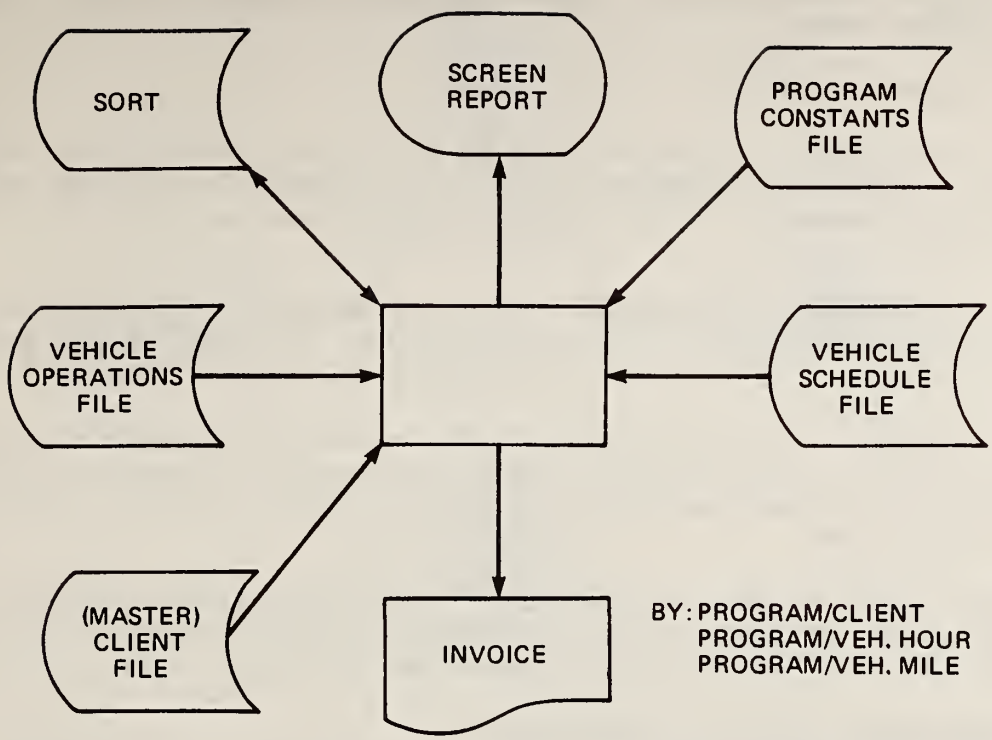
② Scheduler adds PRE-SCHED. TRIPS for date to VEH. SCHED. FILE

③ At appropriate intervals, scheduler adds D-A-R TRIPS to VEH.SCHED. for date: yr/mo/day

④ At appropriate intervals, scheduler prepares DAILY VEH. SCHED. REPORTS (for ea. veh.) for date (copy to driver)

⑤ At conclusion of day of operation, scheduler updates/edits VEH. SCHED. FILE for date.

FIGURE 1-12: AUTOMATED VEHICLE SCHEDULING PROCEDURE



- Provides financial mgmt. staff with standardized reports (invoices) for prescribed periods for program activity in accordance with contractual billing rates.

FIGURE 1-13: AUTOMATED CLIENT/AGENCY BILLING PROCEDURE

- Master Client File: client name, address, and ID number; Title XIX, XX, Medicaid, or other agency number.

- Vehicle Schedule File: client ID number, origin and destination, trip purpose, date of trip.

- Vehicle Operations File: contract vehicle hours, vehicle miles, date.

- Program Constants File: origin/destination codes, mileage, trip purpose codes.

1.2.2.3 Report Generation

Figures 1-14 through 1-17 provide examples of the ways in which the relational DBMS files can be used to combine system data in meaningful ways to support the paratransit manager's decision-making capabilities and reporting requirements.

In Figures 1-14 and 1-15, the daily vehicle operations file and the daily vehicle maintenance and repair file, respectively, are compiled and organized into daily, monthly, year-to-date, or annual overall system reports.

As shown in Figure 1-16, data from the daily vehicle operations file, the vehicle master file, and the daily vehicle maintenance and repair file can be combined to form a fleet operations report for a daily, monthly, year-to-date or annual time frame.

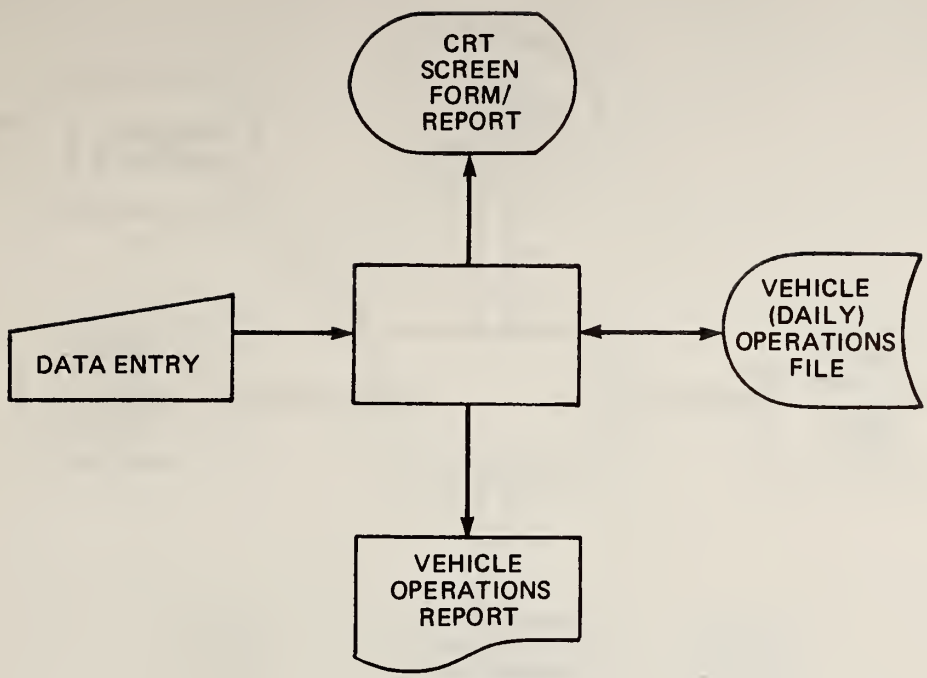
A passenger operations or ridership report can be generated using data from the daily vehicle operations file and the daily vehicle schedule file, again on a daily, monthly, year-to-date or annual basis (Figure 1-17).

1.2.3 Spreadsheet Files and Applications

Microcomputer spreadsheet programs are effective financial management tools. The most useful generic application is in the area of budget preparation and management. The ability of spreadsheets to instantly recalculate an array of values based on a change in one value enables a manager to rapidly update budgetary figures, and to conduct an endless series of "what if" exercises with budgetary projections. Most spreadsheet packages also contain various pre-formatted accounting applications such as payroll; income tax; income and expense statements; cash flow assessment; and accounts payable and receivable.

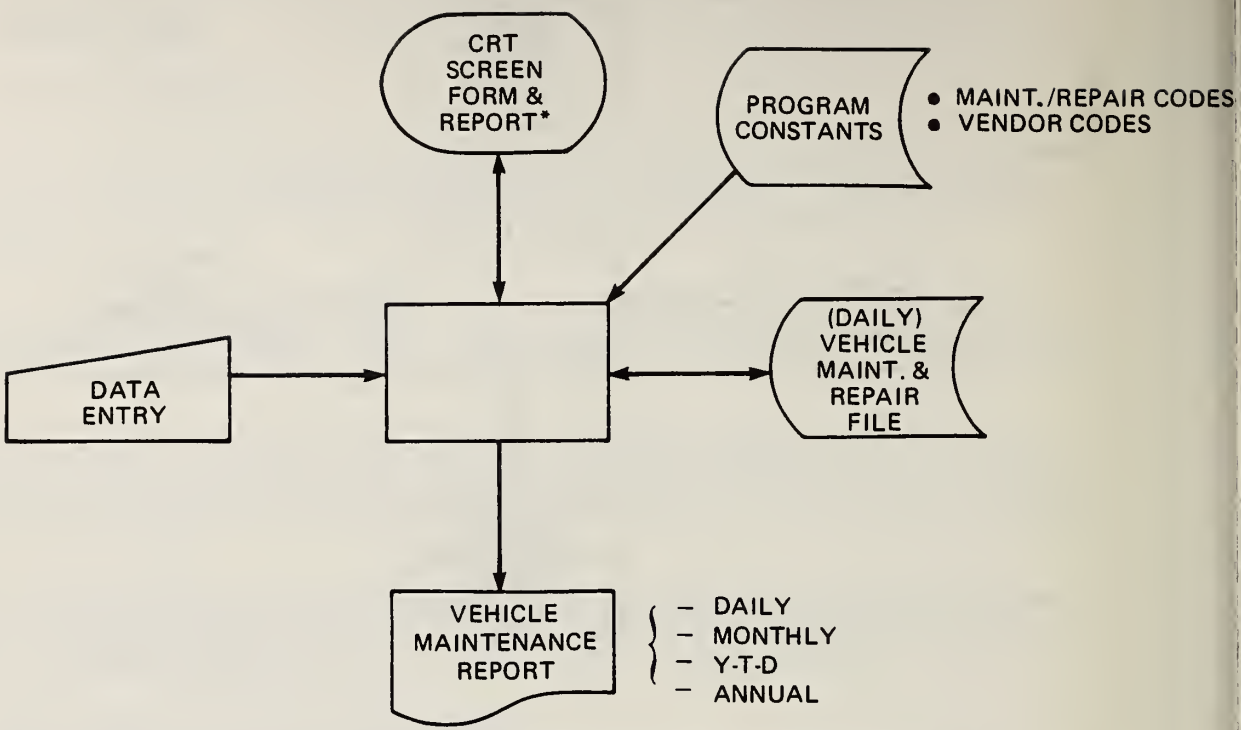
1.2.4 Word Processing and Graphics Files and Applications

Depending on the level of integration existing between the automated system's relational DBMS, spreadsheet, word processing, and graphics programs, reports and presentation materials can be generated from a paratransit agency's operations files with varying degrees of user intervention.



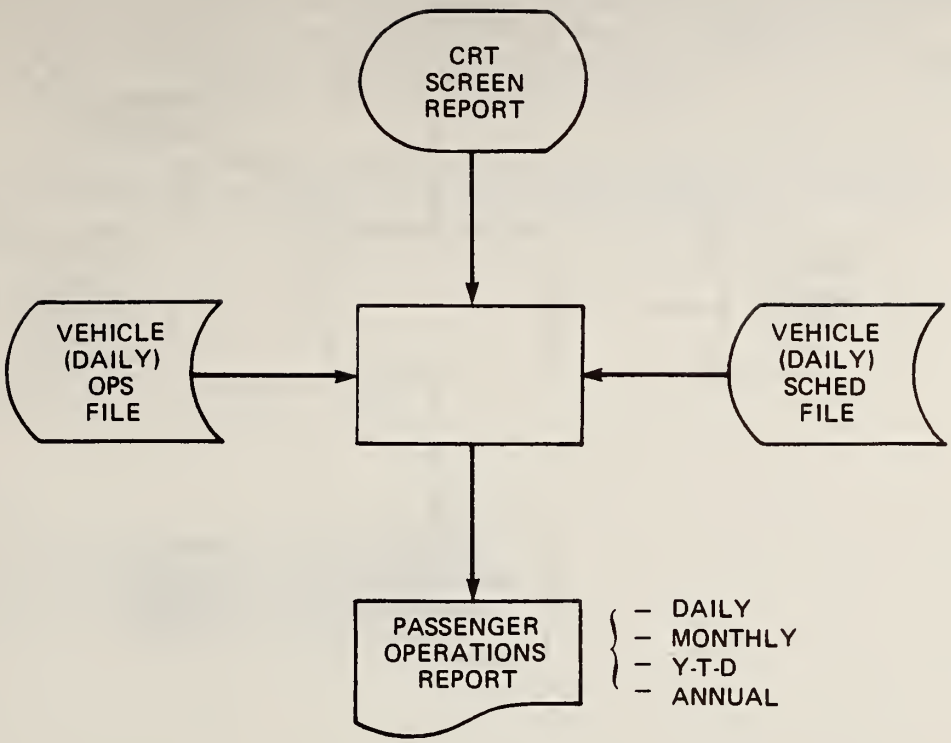
- Allows ops. mgmt. person to build, edit, and generate reports from (DAILY) VEH. OPS. FILE

FIGURE 1-14: AUTOMATED VEHICLE OPERATIONS REPORT



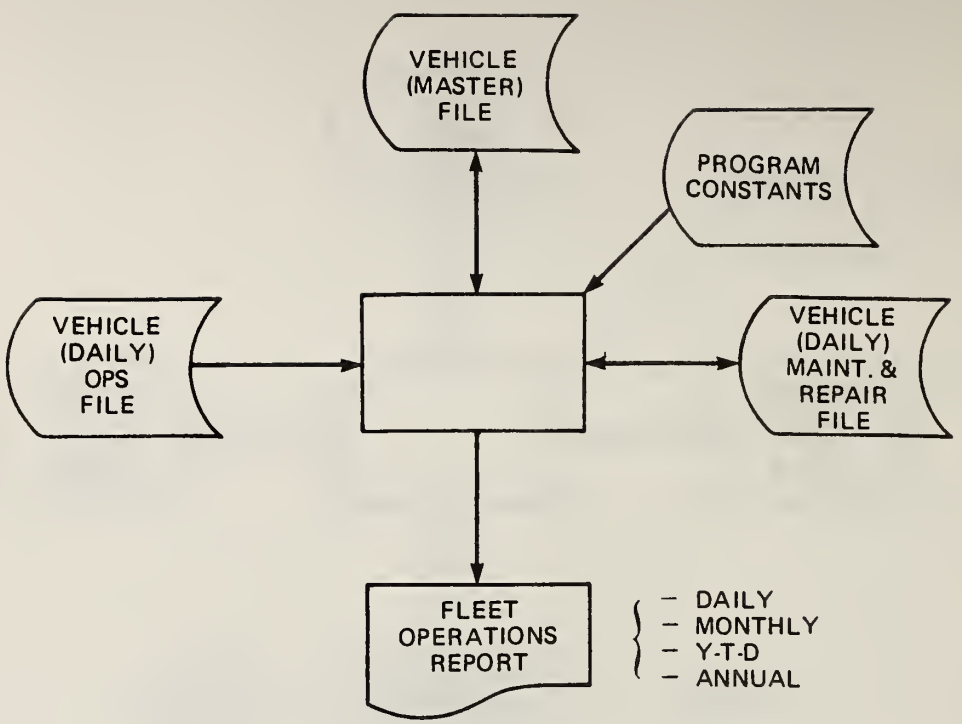
• Maint. person(s) can build, edit, generate report(s) from VEHICLE MAINT. & REPAIR FILE

FIGURE 1-15: AUTOMATED VEHICLE MAINTENANCE AND REPAIR REPORT



- Ops. mgmt. person can generate (daily, monthly, Y-T-D, annual) reports on ridership

FIGURE 1-16: AUTOMATED PASSENGER OPERATIONS REPORT



- Operations mgmt. person can generate (daily, monthly, Y-T-D, annual) reports on ridership

FIGURE 1-17: AUTOMATED FLEET OPERATIONS REPORT

At the highest level of integration, provided by a fully integrated software package, information from data files (such as the daily vehicle schedule file, for instance) can be transferred to the graphics program for organization and graphic presentation. Next, the illustrations can be combined with descriptive text in the word processing program to produce a complete report on vehicle utilization by transportation program for review by funding source administrators or board members.

At the lowest level of integration, provided by separate relational DBMS, spreadsheet, word processing, and graphics programs that are capable of reading and writing files in the same standard format, generation of descriptive graphics with accompanying text requires some amount of user intervention to convert files into the standard format.

The specific files to be maintained in the word processing and graphics programs would depend on the individual paratransit agency's particular reporting needs, but some typical examples are listed below:

Word Processing

- Correspondence File
- Standard Report Format File
- Miscellaneous File

Graphics

- Passenger Operations Graphics File
- Fleet Operations Graphics File
- Financial Management Graphics File
- Program Evaluation Graphics File

2.0 AUTOMATING THE CAR DATA BASE

2.1 INTRODUCTION

This chapter describes the process of using data management and other applications software to automate the management functions of a paratransit agency. The focus is on the data base management software (DBMS). This section discusses the selection of software products; the instructional approach taken in the chapter; the user environment; and the organization of the chapter.

2.1.1 Selection of Software

For the task of automating the CAR data base we have selected Microrim's relational data base manager R:base 4000. As discussed in Section 1.0, the advantage of a relational data base compared to a traditional file management system is the capability of "relating" different files. For example, data from two different files can be joined together to create a third file. Also, data records in separate files with common fields of information can be updated or deleted simultaneously.

While we are highly enthusiastic about the potential applications of relational data managers to transit management, this document does not in any way constitute an endorsement of a specific product. In using R:base 4000 to automate the case study data base, we engaged in a learning process in which we discovered the product's strengths and weaknesses. We discuss both candidly in this manual and provide suggestions for overcoming the weaknesses and taking advantage of the strengths. Funding limitations prohibited similarly intense examinations of competitive products.

The selection of a data base manager is not a decision to be taken lightly. They can cost anywhere from $400 to $800 and there is no money back guarantee. The software industry is highly volatile. New, innovative products are announced almost weekly. Products are sent into the marketplace with bugs which do not become apparent until field testing. Each product has unique strengths and weaknesses which require that the selection process be responsive to the specific needs of the transit agency.

We strongly urge any transit agency interested in using a data base manager to conduct their own selection process and evaluation. To assist agencies in doing so, we have provided in Appendix A of this manual suggestions for how to conduct a software procurement process; criteria to utilize in evaluating data base management software; and information on a variety of competitive products which you might want to consider. While this document is clearly written to a specific piece of software (R:base 4000), we believe that the generic approach described herein will be helpful to users interested in applying any data base management software to paratransit management.

In addition to the data base management software itself, this section will also discuss the integration of data base management with other software applications such as spreadsheets, word processing, and graphics.

For these other functions, we have used Perfect Calc, Perfect Writer, and Fast Graphs. Again, our use of these products does not constitute an endorsement over the many similar products in the marketplace. In addition, the use of this software in conjunction with R:base 4000 does not constitute a recommendation as to the effectiveness of these products as an integrated package. To the contrary, Perfect Calc is not one of R:base's recommended spreadsheets and integration is in fact cumbersome.

2.1.2 Instructional Approach

The purpose of this chapter is to assist the professional transit manager in applying a data base management system to his or her own management problems. It is NOT a replacement for the manual and tutorial which accompanies the software. Rather, it is intended to function as a bridge between the manager and the software manual. Software manuals are written for generic applications. This manual is written for the sole purpose of applying R:base 4000 to paratransit management. In doing so, we provide both a conceptual framework and specific instructions for using R:base 4000. These instructions are intended as examples and are not all inclusive. We have not replicated the R:base manual. We do think we have made it easier to use and to apply to this specific function.

The authors of this document are transit professionals who have learned to use applications software to solve transit management problems. We are NOT computer professionals. We have written this manual for people like us.

In writing instructions for R:base 4000, we assume that the audience has little or no experience in the use of data base management software. We do assume, however, that the user is familiar with common data processing terms and with the operation of their hardware system. We hope that in combination, this manual and the R:base manual will enable a transit manager to perform the basic functions of R:base 4000. Some of the more complex functions may require greater data processing expertise. In any event, the user will be on much firmer ground in understanding the functions which R:base can perform, and working with a data processing professional in tailoring the system to their needs.

The focus of this document is clearly on the application of data base management software. This type of software is the most complex to use among the most widely applied software. Therefore, our treatment of the other applications software discussed in this manual (spreadsheets, word processing and graphics) is more cursory and assumes that the basics of using these programs can be mastered by following the manuals and tutorials which accompany them. Somewhat more detail is provided on spreadsheets than on the other two applications. Our emphasis will be on the application of this software in conjunction with a data base manager.

A final point which needs emphasis is that the case study data should be viewed as hypothetical. The reasons for using data from an actual transit system were to develop a realistic case study, and to avoid the necessity of developing a data base from scratch. This data dates from 1979 and was generated by a unique transit system operating in a unique environment. The user should concentrate on the applications, and not the data.

2.1.3 User Environment

R:base is available for use on a variety of hardware and operating systems. Operating systems include the following:

- MS-DOS Release 1.1 (or higher)
- PC-DOS Release 1.1 (or higher)
- CTOS Release 8.0 (or higher)
- BTOS Release 8.0 (or higher)

R:base requires 256K bytes of memory for operation. Hardware with 128K can be easily expanded at reasonable cost to 256K by means of expansion boards. We developed this prototype on a Columbia VP microcomputer using MS-DOS 2.0. R:base can utilize any 80 or 132 column ASCII printer compatible with your hardware. We recommend a printer with a 132 column capability given the types of reports you are likely to produce. The printer which we employed is an 80-column printer with an optional 132-column small type. This accounts for the different size printing in some of our outputs.

One of the major selling points of R:base is its tremendous capacity potential. This is illustrated in Table 2-1.

TABLE 2-1. DATA BASE SPECIFICATIONS

Maximum number of files per data base	40
Maximum number of fields per data base	400
Maximum record size	1530 characters
Maximum records per file	2.5 billion*
Maximum records per data base	100 billion*
Maximum command line input	1600 characters

*or limited by the file size of your operating system

It is the rare paratransit system indeed which would run up against the limits of R:base's theoretical capacity. Of more practical concern is how to tap this capacity. This prototype was developed using two floppy disk drives. Since the R:base diskette must occupy one drive at all times, and has limited writing space, the user is limited to the capacity of a single diskette at any one time. We found this capacity sufficient to build a one-week data base for a five-vehicle system. It might have been possible to go as high as two weeks. Ideally, records should be stored in monthly increments to coincide with typical scheduling, invoicing, and recordkeeping procedures. This would not be feasible using R:base for more than a 1-3 vehicle system (depending on ridership levels).

There are two solutions to this problem. One is to store data on multiple floppy disks. We found this process to be cumbersome and do not recommend it. The alternative and recommended solution is to use a hard disk. A recommended hard disk has a capacity of 20,000,000 (20 megabytes) bytes of memory compared to 360,000 for a floppy disk. While expensive, prices are

falling and represent a modest increment to the overall cost of your system. In the course of this manual, we do provide guidance on operating with floppy disks. Appendix B provides more detailed space management instructions.

Another of R:base's advantages is its user friendliness. Most data entry can be accomplished through a series of prompts which clearly call for specific pieces of data. Many user decision points provide menus of clearly defined choices (hence, the term menu-driven). Commands can be given in standard (if somewhat stilted) English sentences.

Due to these features, R:base can literally be taken off-the-shelf and used as is. However, transit systems with a large number of data entry clerks may desire additional customization to further simplify and standardize data entry. An example of customization is provided in Appendix C.

This Manual is based on R:base 4000 version 1.01. However, during the latter stages of this project (April 1984), an updated version (1.11) was issued by the manufacturer. The new version is completely compatible with the version used in this Manual. It also includes several improvements and enhancements. The customizing example used in Appendix C is based on the newer version as these features were not available in the old version. In addition, Appendix F provides a brief description of other program improvements.

2.1.4 Organization of Chapter

The remainder of this chapter is organized into four parts. Section 2.2 describes how to begin working with R:base and setting up a data base. Section 2.3 provides a description of how to use R:base to accomplish the day-to-day scheduling function. Section 2.4 explains how to generate end-of-the-month reports. Section 2.5 discusses the application of spreadsheet, word processing, and graphics software.

Throughout this chapter, examples are displayed of screen forms, command formats, and reports. Appendix D contains all such formats, illustrating how to set-up an entire data base for paratransit management. Appendix E contains blank copies of format sheets for reproduction and use in your own agency.

2.2 USING R:BASE TO SET-UP A DATA BASE

This section provides information on the following:

- The organization of the R:base manual
- On-Screen Help provided by R:base
- The structure of R:base
- Conceptualizing a Data Base
- Constructing a Data Base
- Entering Data

Several conventions are followed throughout the remainder of the manual. Information which must be input into the computer by the user appears either within a figure or in quotation marks. Except for one occasion which is clearly distinguished, the user does not input quotation marks. They are present for clarification only. If a generic term is used to which the user must apply a specific value (i.e., attribute name), the generic term is underlined. Information which the user may optionally input appears in parentheses. Computer-generated responses which will appear on the user's screen are typed in **bold face.**

2.2.1 The R:base Manual

The R:base Manual has several important features of which the user should be aware at the start of the process. These include the following:

- A tutorial provides both written documentation and a diskette which takes the user through the development of a small data base. This tutorial is available separately for a small fee and provides the user with a good sense of the operation of R:base. The diskette contains a small data base on which the user performs a series of functions. One weakness in the tutorial is that the example does not complete a full cycle back to the starting point. Thus, a second employee would not be able to simply pick it up and run through it from the start. The initial user must acquire sufficient familiarity with R:base to return the data base to its original format for others to use.

- An insert at the front of the manual discusses quirks in the software and changes which have been made since the last complete update of the manual. We recommend that the user write these changes directly into the body of the text.

- The body of the manual containing the following sections:

  - Introduction
  - Data Base Structure
  - Data Input
  - Data Inquiry
  - Reports
  - Data Modification
  - Relational Operations
  - Transportability
  - Customization

- An appendix providing an extremely useful summary of R:base commands.

- An appendix containing a glossary of commonly used terms.

- An appendix providing explanations of both generic and command-specific error messages.

- An appendix providing instructions on the use of the RBEDIT function for generating and editing ASCII command and data files.

- An appendix containing instructions for starting and installing R:base under a specific operating system.

- An index.

2.2.2 On-Screen Help

In addition to its extensive written documentation, R:base also contains two on-screen assistance formats. These are the "help" and "prompt" modes. The help mode provides on-screen descriptions by command. Prompt is designed to assist the user in formatting commands and data entry. Section 2.2.3 describes how to access these modes.

We found that these modes are helpful for the new user. However, after two weeks of intensive usage, we found that the "help" mode had very little to tell us that we didn't already know (which isn't to say that we weren't having problems!). We also found the "prompt" mode to be a cumbersome method of data definition and entry compared to other modes which will be discussed below. This was particularly true of large (yet typical) files.

2.2.3 How R:base Works

2.2.3.1 Structure

There are four key terms involved in understanding the structure of R:base:

- Data Base
- Relation (file)
- Attribute (field)
- Row (record)

These elements are displayed graphically in Figure 2-1.

The basic unit of organization is the data base. All related information must be housed within a single data base! There is no interaction between different data bases. For most paratransit agencies, it should be possible to house all related data in one data base, given that a single data base can hold 100 billion records in 40 separate files.

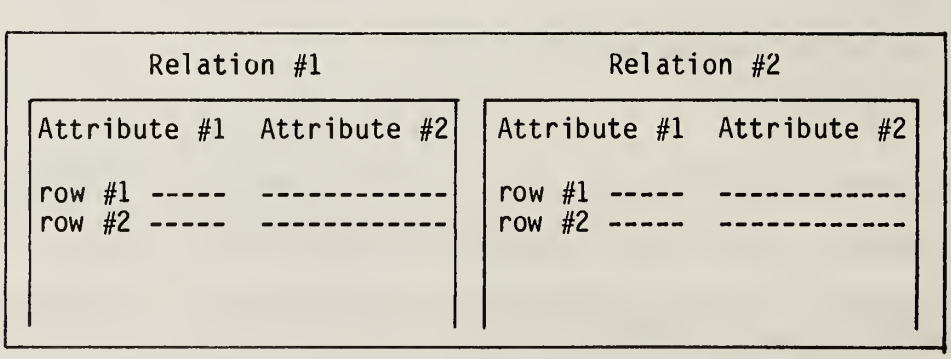


FIGURE 2-1. THE STRUCTURE OF R:BASE 4000

A relation is synonymous with a data file. A single relation contains a set of closely interrelated data which the user would file together and keep separate from other information in a manual system. Examples include a client file and a vehicle schedule file. Each R:base data base can contain up to 40 relations.

An attribute is synonymous with a single field of data in a relation (file). For example, client ID number and last name would be typical attributes (fields) in a master client relation (file). Each data base can contain up to 400 attributes. Each attribute can be associated with more than one relation permitting integration and update across relations (files). This is a major advantage over the file manager type of software.

A row is a single data record across a number of attributes in a relation. For example, all the data in the master client relation pertaining to any one client constitutes a row. R:base has a theoretical capacity of 2.5 billion rows per relation, and 100 billion rows per data base.

2.2.3.2 Entering/Exiting

Entering R:base requires two distinct actions (which are contained on 2 separate diskettes in the floppy version). Type "rbase" in response to your operating system prompt. The rbase logo (a giant "R") will appear on the screen with instructions to insert the second diskette and then press any key. This will cause the program to load. When the program has loaded, you will receive the R:base prompt shown in Figure 2-2.

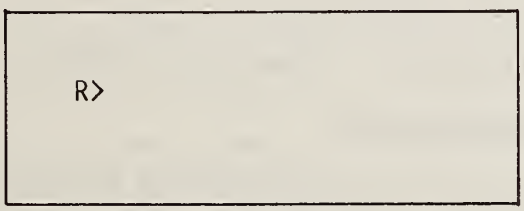


FIGURE 2-2. R:BASE PROMPT

To exit R:base, simply type "exit" in response to the R:base prompt. This will return you to the disk operating system. No command is necessary to save input. All input is automatically written to disk and saved.

2.2.3.3 Modes

Once inside R:base, you have the option of entering one of four special function modes, or of remaining in the main R:base module. The four special modes are as follows:

- Help: The help mode provides on-screen instructions for the operation of R:base. To enter, type "help" for an overview of all R:base commands, or "help command name" for instructions regarding a specific command.

- Prompt: The prompt mode provides menu-driven choices for performing R:base functions. To enter, type "prompt" for a listing of all commands, or "prompt command name" for a specific command prompt.

- Define: The define mode must be entered to establish or modify your data base. When establishing a new data base, enter "define data base name". When modifying an existing data base which you are already inside, simply enter "define". A detailed description of the process for establishing a data base is contained in Section 2.2.5.

- Load: The load mode is one of four methods for entering data into R:base. We found it to be highly inefficient except for very small data files. We will focus on what we consider to be the preferred method of data loading in Section 2.2.6. To enter, type "load".

Upon receiving the appropriate entrance command, R:base will respond with a new prompt which will be the first letter of the mode you have entered. Thus, users can always tell what mode they are in. Figure 2-3 displays the possible R:base prompts. To leave a mode and return to the general R:base system, type "end". (Note that to leave a mode, you type "end". To leave R:base entirely, you "exit".)

```
R> (general R:base system)
H> (help)
P> (prompt)
D> (define)
L> (load)
```

FIGURE 2-3. R:BASE MODE PROMPTS

2.2.3.4 Editing and Syntax Messages

You can use your screen editor to correct typing mistakes. An incorrectly entered command will receive a syntax message displaying the correct command format as shown in Figure 2-4.

```
                              SYNTAX

The syntax for the SELECT command is:

   SELECT (attname1 attname 2 ...) FROM relname (SORTED BY ...) (WHERE)

Attribute types may have "=s" appended for summations or "n" when
specifying a display width
```

FIGURE 2-4. SYNTAX MESSAGE

The syntax message will be followed by the appropriate prompt to enable you to resume input. We did find on a few occasions that grievous input errors were

unrecognizable to the syntax error response system and bombed out the program. While no data was lost, it was necessary to reload the program and start again.

2.2.3.5 Special Functions

There are several functions with which the user will need to become familiar for the efficient operation of R:base. These include the following:

- Set: The set command enables the user to alter a variety of default procedures. For example, line width can be changed from 80 columns to 132 simply by entering "set width 132". Output can be sent to the printer instead of the terminal by entering "Output printer". All default values return after exiting R:base and must be changed again in the next session.

- List: There are several "list" commands which enable the user to view the structure of the data base which is being created. These commands include the following:

  - Listrel: Lists the names of all relations
  - Listatt: Lists the names and characteristics of all attributes
  - Listrel all: Lists all relations and all features of each relation except rules

- Show: The show command enables the user to view the status of functions subject to the set command. It can also display any rules which the user has established for the data base (see Section 2.2.5).

- Long Commands: A plus sign (+) is used to continue a command which exceeds 80 columns in length. Before reaching the end of the line, enter a blank space and then "+". Hit return and continue the command on the next line.

- Multiple Commands: A semicolon (;) can be used to separate multiple commands entered on the same line.

- Reusing the Previous Line: R:base always remembers the last line, whether a command or data entry. You may reuse one or more items from your last entry by using the following notations:

Notation	Meaning
*	Causes the corresponding input field to be repeated
*n	Causes the corresponding next "n" items to be repeated
**	Causes the corresponding item and all remaining items to the end of the line to be repeated

2.2.4 Conceptualizing the Data Base

Prior to initiating any work on the computer, the user should first outline manually the structure of the data base which will be created. This process involves the following steps:

- Determine what relations (files) will be needed to contain your data in a logical fashion.

- Assign attributes (fields) to each relation, paying particular attention to the potential for using common attributes across relations, and determine which are the key attributes for accessing purposes.

- Determine what rules, if any, will be employed to control the consistency and accuracy of data entry.

- Determine where passwords will be necessary to provide security for the data base.

In this prototype, we have created the following relations:

- Master Client File ("msclient") provides a permanent record of each client.

- Pre-Schedule Trip File ("prsched") provides a listing of regularly scheduled trips on a monthly basis. This relation should provide sufficient information for dispatchers to use in scheduling and for drivers to use in running routes.

- Real Time File ("realtime") contains the same format as the pre-schedule file, but will contain only true dial-a-ride trips as they are requested.

- Vehicle Schedule File ("vsched") contains all trips scheduled during the course of a month by combining the preschedule and real time files.

- Vehicle Operations File ("vehops") contains information on daily vehicle operations such as miles travelled, service hours, fuel and maintenance expenses.

- Master Vehicle File ("vmaster") is a permanent record of each vehicle operated by the agency.

The order in which relations are established is not important, although typically you would begin with the master client and vehicle files. All command and screen formats needed to create each prototype relation are contained in Appendix D. Section 2.2.5 will provide a narrative description of how to create the vehicle schedule files. Section 2.2.6 will describe how to enter data into these files. Section 2.3 will explain how the two vehicle schedule files will interact to assist the agency in the performance of the vehicle scheduling function.

2.2.5 Creating a Data Base

This section provides a step-by-step description of the creation of a data base. All commands used to establish the sample relation are summarized in Figure 2-14 at the end of the section. The process of creating a data base is summarized in Figure 2-5 below:

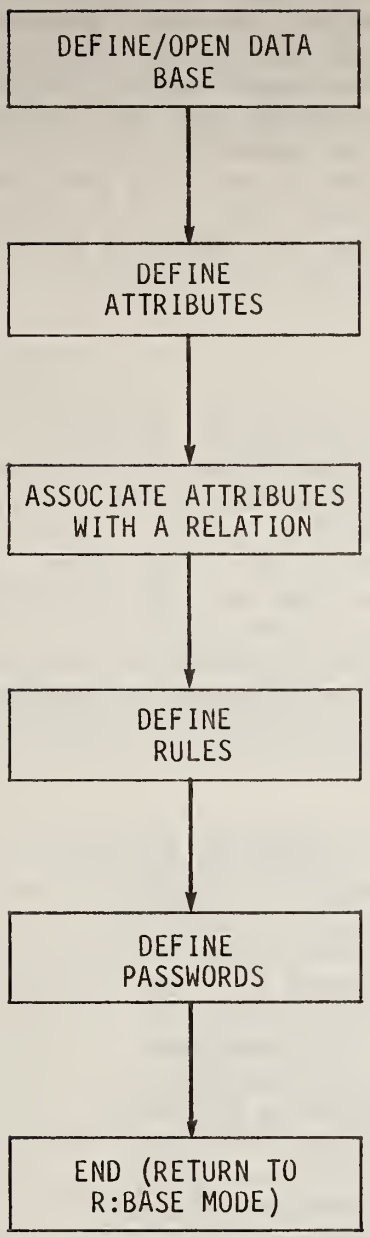


FIGURE 2-5. CREATING A DATA BASE

2.2.5.1 Accessing a Data Base

The first step in creating a data base is to give it a name. We have called our data base "CAR" (short for Call-A-Ride of Barnstable County). All names in R:base (data bases, relations and attributes) are limited to a maximum of eight characters, and may not contain blanks. If you are using the floppy disk version, you will want to create your data base on the "b" drive, since the R:base diskette occupies the default, or "a" drive. To input on the "b" drive, you must enter "b:" prior to the data base name. This leaves only six more characters for the data base name ("b:" is essentially read as part of the name). Thus, we established our data base by entering the following:

```
R> "define b:car"
D>
```

R:base will respond with the D> prompt, indicating that you are now in the define mode. To enter this data base in the future, you will type "open b:car". If everything is alright, R:base will respond **Data Base Exists.** To alter the definition of a pre-existing data base, simply enter "define" after you have opened the data base.

Everything you do in R:base will be done in the context of a single data base. Therefore, each session at the computer must begin with the command to either open an already existing data base, or define a new one. If you enter anything else, R:base will respond **Data base does not exist.** Thus, using R:base should be thought of as a three-tiered process as displayed in Figure 2-6.

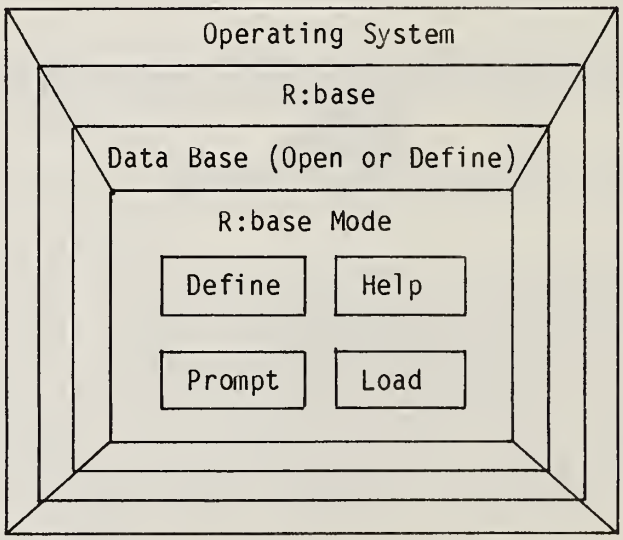


FIGURE 2-6. R:BASE TIERS

First, you enter R:base from the operating system by loading the disks. You can do only one of two things at this point, open or define a data base. Once done, you can proceed to the inner core where you can remain in the general

R:base mode to perform a variety of functions, or enter one of the four specialized modes. Two additional modes (forms and reports), will be introduced later.

After defining a data base, you will want to create a password to limit access to it. Enter "owner password name" in response to the define prompt (D>). In order to access the data base in the future, you will need to enter "user password name" after opening the data base and receiving the **Data Base Exists** response. We have defined the following password:

```
R> "Owner Marc"
```

Keep track of your passwords. In the interest of security, R:base can't tell you what they are.

2.2.5.2 Establishing Attributes

Once inside the define mode, the first step will be to define attributes (fields). You can define all the attributes which you will use in the entire data base, or only those which will be associated with one specific relation. Additional attributes can be added to the data base at any time and associated with whichever relations are appropriate. We recommend establishing one relation and its associated attributes at a time.

Before sitting down at the computer to define attributes, we recommend laying them out manually first, as shown in Figure 2-7. Each record in this relation will have a potential value for each one of these attributes. A blank form has been provided in Appendix E for the user to reproduce and employ in their work.

Attribute	Name	Type	Length	Key
Trip Date	Tripdate	date		
Vehicle Number	veh#	text	2	
Client ID	clientid	text	4	key
Last Name	lastname	text	15	
First Name	frstname	text	10	
Pick-Up Time	putime	integer		
Address	address	text	15	
City	city	text	10	
Destination	destinat	text	15	
Number of Trips	trip#	integer		
Round Trip ?	rt	text	1	
Round Trip Time	rttime	integer		
Passenger Classification	paxclass	text	3	
Trip Purpose	trippurp	text	3	
Payment Code	paycode	text	4	
Special Needs	specneed	text	10	
Edit Date	editdate	date		

FIGURE 2-7. ATTRIBUTE DEFINITION FOR VEHICLE SCHEDULE FILES

Each attribute definition contains four possible elements:

- Name - This is a maximum eight-character label which you give to each attribute. The name should make sense to you and be easy to remember.

- Type - There are six attribute types. Simply enter the name of the appropriate type as follows:

  - Date: Dates are entered in a default format of mm/dd/yy. This can be changed to any other combination through the set command.

  - Dollar: Represents dollar amounts (assumes two decimal places).

  - Integer: Represents whole numbers up to nine places.

  - Time: Represents the time in hh/mm/ss format. Twenty-four hour military time must be used.

  - Real: Represents whole numbers with decimals.

  - Text: Represents everything else (all alphanumeric entries).

- Length - The number of columns set aside for each attribute's data entry. The user defines length only for text attributes. All other attribute lengths are established by default. The text default, requiring no user entry, is "8" characters. To enter a length, simply type in the number of columns desired.

- Key - Keys identify attributes through which the user desires to access data. The use of key attributes is OPTIONAL. In R:base, data can automatically be accessed through ALL attributes. The only advantage of a key attribute is that it accesses data faster. The only disadvantage of using keys occurs if you are using floppy disks. Keys occupy substantial disk space which you can ill-afford in the floppy disk mode. If your data base is small enough to warrant using floppy disks, it is doubtful that you need the speed advantage of keys. (In Appendix B, we discuss how to calculate required disk storage space.) We have designated a single key attribute, "clientid", throughout the CAR data base.

The attributes composing the two vehicle schedule files (pre-scheduled trips and real time), are displayed in Figure 2-7. Most of the information is self-explanatory. Client ID number has been established as the key field. Depending on how the user most frequently accesses data, last name might be a better choice for a key field.

Round-trip data is included for two reasons. Including both trips on one record greatly reduces data entry time and disk storage space. Since all the other data is the same, why enter it twice?

The "number of trips" attribute is essentially a "counter" for producing reports which can analyze how many trips were taken in various categories.

This process is explained in Section 2.4. The number entered in this attribute would normally be "1" (one-way trip) or "2" (round-trip), unless the client was accompanied by an escort.

"Paycode" enables you to tag each trip record for later billing purposes. "Specneed" provides any unusual information about the client (medication, structural problems at the home, disabilities, etc.) which a driver will need to know.

Our selection of attribute "types" requires some explanation. Although certain entries may consist of all numerals and be thought of as an "integer" or "real", (such as vehicle number and client ID), we have entered them as "text". Text has the advantage of allowing the user to define field length (and thus limiting data base size to the minimum required to receive the data). There is no disadvantage in doing this so long as you do not plan to perform any arithmetic function on the data. Although client ID is a number, you would never add two client ID's together. Thus, they can be safely called text fields.

We have also chosen to define the time fields as "integers" instead of "times". The reason is unique to R:base which defines time as hours:minutes: seconds. It is the rare transit authority which schedules vehicles to the second. If the time definition is used, seconds (even "00") must be entered. This imposes an unnecessary data entry burden and increases the size of the data base. By using "integer" we can also eliminate the colons. Thus, 10:00 a.m. is entered as "1000".

To begin defining attributes, type "attributes" in response to the "D>" prompt (note the common sense nature of most R:base commands). Then enter one attribute to a line exactly as shown in Figure 2-7, leaving one blank space between each item in the definition.

We strongly urge you to examine each attribute definition carefully before moving on to the next line. You cannot go back to correct a previous line. The process of revising attributes is extremely poor and is a major weakness of R:base. Similarly, you should know what attributes you intend to include in each relation before starting. The process of revising data base structure is defined in Section 2.2.5.8.

R:base may contain records which have up to 1530 columns. It is unlikely that you will want to create a record of anywhere near this length. You should keep in mind that you can never view more than 132 columns on the screen or in print at any one time. If you have a standard 80 column screen, the remaining columns will be wrapped around into a second line. Wide carriage printers can print 132 columns across. Some 80 column printers have a small type mode which prints 132 columns. You will usually not need to see all attributes at once. If you would like to be able to do so, the attribute lengths which you associate in a relation cannot exceed 132 columns. Each attribute type has the following default length:

- Date 8
- Dollar 21
- Integer 9
- Real 16
- Text 8
- Time 8

In counting up total attribute length, remember that if the attribute name exceeds the length of the attribute field, you must add the columns occupied by the name which will appear horizontally as headings at the top of the screen or page. Thus, although Client ID has a data field length of 4 columns, the label "clientid" occupies eight columns.

While you most likely won't need to see all attributes at any one time, certain attributes are likely to be called up frequently. For example, in scheduling, you need to see pick-up time, address, destination, return-time, etc. You should carefully consider the use to which the relation will be put before defining its attributes. Any attribute which you will need to call up frequently and quickly should be located in the first 132 columns of the relation. See Section 2.3.5 on scheduling dial-a-ride trips for further discussion of this issue. The customizing features described in Appendix C provide further suggestions on overcoming these limitations.

2.2.5.3 Defining Relations

Upon completing attribute definition, enter "relations" to initiate relation definition. On the next line, enter the name of your relation and the attributes with which it is associated as shown in Figure 2-8.

```
D> relations
D> prsched with tripdate veh# clientid lastname +
D> frstname putime address city destinat trip# rt +
D> rttime paxclass trippurp paycode specneed editdate
```

FIGURE 2-8. DEFINING A RELATION

You must use the word "with" between the relation name and the first attribute.

We have now created a relation called "prsched", associated with 17 attributes. Note the use of a plus sign (+) to continue the definition process across three lines. Attributes can be associated with a relation in any order. However, the order in which the attributes are listed in Figure 2-8 is the order in which data entry will take place. We therefore recommend listing the attributes in an order which makes logical sense for data entry.

2.2.5.4 Defining Rules

Data entry rules are a valuable tool for controlling the accuracy of the data entry process. To begin rules definition, enter "rules". On the following lines, enter your rules in one of the following two formats displayed in Figure 2-9.

```
(1)  "Error Message" attribute name (in relation name) +
     EQ value (and/or attribute name ----)
     NE
     GT
     GE
     LT
     LE
     CONTAINS
     EXISTS

(2)  "Error Message" attribute name 1 in relation name +
     eqa attribute name 2 in relation name and/or
     nea
     gta
     gea
     lta
     lea
```

FIGURE 2-9. RULES DEFINITION FORMATS

In example (1), an attribute is being measured against a specific value. In example (2), two attributes are being measured against each other. The relation name in example (1) is optional (hence it appears in parentheses) and is only required if the attribute is associated with more than one relation. The error message should be clearly understandable to the data entry personnel. It must be entered in quotation marks. This is the only input form which requires quotation marks. Figure 2-10 defines the arithmetic operator abbreviations used in rules definitions. Figure 2-11 illustrates the rules commands entered for the relation "prsched" for the prototype data base.

```
EQ = equals = EQA
NE = not equal = NEA
GT = greater than = GTA
GE = greater than or equal to = GEA
LT = less than = LTA
LE = less than or equal to = LEA
CONTAINS = contains a test value
EXISTS = field must contain data
```

FIGURE 2-10. RULE OPERATORS

```
"Passenger code is incorrect" paxclass eq e or paxclass +
eq eh or paxclass eq h or paxclass eq weh or paxclass eq wh

"Trip code is incorrect" trippurp eq hc or trippur eq nu +
or trippurp eq mow or trippurp eq se or trippurp eq ft +
or trippurp eq gt or trippurp eq adc or trippurp eq dl

"Trip time is too early!" putime ge 0700

"Round trip code is incorrect" rt eq y or rt eq n
```

FIGURE 2-11. DEFINING RULES FOR CAR

In the example shown in Figure 2-11, a data entry clerk will be prevented from entering incorrect codes for passenger class, trip purpose, or round-trip. In the latter case, the data must read either "y" (yes) or "n" (no). Figure 2-12 displays the definitions of the two other codes which we have created for this relation. Similarly, a dispatcher will be prevented from scheduling a trip before 7:00 a.m. Should an incorrect entry be made, the designated error message will appear on the screen.

Passenger Classification	Trip Purpose
e = elderly	hc = health care
h = handicapped	nu = nutrition
eh = elderly/handicapped	mow = meals on wheels
wh = wheelchair handicapped	se = special education
weh = wheelchair elderly handicapped	ft = field trip
	gt = group trip
	adc = adult day care
	dl = dialysis

FIGURE 2-12. VEHICLE SCHEDULE CODES

One should note that error messages are typically stated as negative conditions (i.e., something is incorrect) to inform the user that a mistake has been made. However, generating the message requires a series of positive conditions. In the first example in Figure 2-11, you must "tell" R:base what the correct conditions are to enable it to recognize a mistake. Thus, the string of conditions following the error message should be read as "paxclass (must) equal e", etc.

The rules command displays two important features of all R:base string type commands. First, the maximum number of conditions which can be strung together is ten (10). Second, the command language is a little stilted in that attribute names must be repeated for each condition. Whereas one might in normal conversation say that paxclass must equal e, h, eh, wh, or weh, in R:base one must repeat "paxclass" for each condition. Failure to do so will generate an error and syntax message.

In stringing several conditions together, you must exercise great care in the proper use of "and/or". The use of "and" creates a series of conditions all of which must be satisfied or an error will be generated. The use of "or" creates a series of conditions of which only one need be true. "And" and "or" conditions can be combined within a single string of conditions.

2.2.5.5 Passwords

The final step in database definition is assigning passwords to specific relations (as opposed to the data base as a whole). There are two kinds of relational passwords, RPW (read passwords) and MPW (modify passwords). Read passwords permit the user to read data but not modify it. Modify passwords permit the user to both read and modify data. Passwords are entered in the following format:

```
D> Passwords
D> RPW for prsched is Tony
D> MPW for prsched is Patti
```

FIGURE 2-13. DEFINING PASSWORDS

2.2.5.6 Ending Data Base Definition

Following password definition, enter "end" to leave the define mode and return to the general R:base module (but still within the data base which you have just defined). If there are any attributes which have not been associated with a relation, you will be warned by R:base that the attribute will be lost if you leave the define mode without associating it with a relation. For this reason, we recommend defining attributes for one relation at a time. If there are no problems, you will receive the response **End R:base Data Base Definition.**

Figure 2-14 displays the complete process of establishing the data base "CAR" and the relation named "prsched". You can check on the data base which you have just created by entering "listrel prsched". Figure 2-15 displays the hardcopy print-out of the response. You can view all the rules created for a database by entering "show rules". Figure 2-16 displays the response. Rules 6, 7, 8, and 9 were created for the relation "prsched".

Whenever you end a major step, such as defining a relation or entering data, we recommend that you exit R:base and make a copy of your data base using the copy function of your operating system. Be sure to copy the three *.RBS files associated with your data base. See Appendix B for a description of R:base file structure.

2.2.5.7 Creating Additional Relations

Additional relations can be created within the CAR data base by following the procedure described above. Simply "open b:car" and enter "define". Your next step depends on the attributes of the relation which you are planning to create. For example, the relation "realtime" will have the exact same

```
R> Define b:car
   Database exists
D> owner marc
D> attributes
D> tripdate date
D> veh# text 2
D> clientid text 4 key
D> lastname text 15
D> frstname text 10
D> putime integer
D> address text 15
D> city text 10
D> destinat text 15
D> trip# integer
D> rt text 1
D> rttime integer
D> paxclass text 3
D> trippurp text 3
D> paycode text 4
D> specneed text 10
D> editdate date
D> relations
D> prsched with tripdate veh# clientid lastname frstname putime +
   address city destinat trip# rt rttime paxclass trippurp paycode +
   specneed editdate
D> rules
D> "Passenger code is incorrect" paxclass eq e or paxclass eq eh +
D> or paxclass eq h or paxclass eq weh or paxclass eq wh
D> "Trip code is incorrect" trippurp eq hc or trippurp eq nu or +
D> trippurp eq mow or trippurp eq se or trippurp eq ft or trippurp +
D> eq gt or trippurp eq adc or trippurp eq dl
D> "Round trip code is incorrect" rt eq y or rt eq n
D> "Trip time is too early!" putime ge 0700
D> passwords
D> rpw for prsched is Tony
D> mpw for prsched is Patti
D> end
   End R:base Data Base Definition
R>
```

FIGURE 2-14. DEFINING PRSCHED

```
Relation: prsched
Read Password: YES
Modify Password: YES

        Attributes
  # Name      Type      Length          Key
  1 tripdate  DATE         1 value(s)
  2 veh#      TEXT         2 characters
  3 clientid  TEXT         4 characters yes
  4 lastname  TEXT        15 characters
  5 frstname  TEXT        10 characters
  6 address   TEXT        15 characters
  7 city      TEXT        10 characters
  8 paxclass  TEXT         3 characters
  9 specneed  TEXT        10 characters
 10 putime    INTEGER      1 value(s)
 11 destinat  TEXT        15 characters
 12 rt        TEXT         1 characters
 13 rttime    INTEGER      1 value(s)
 14 trip#     INTEGER      1 value(s)
 15 trippurp  TEXT         3 characters
 16 paycode   TEXT         4 characters

        Attributes
  # Name      Type      Length          Key
 17 editdate  DATE         1 value(s)
```

FIGURE 2-15. PRSCHED RELATION

```
     RULE checking = ON
RULE   1     zip ge 0200
        and zip le 02999
    Message:Zip code is incorrect
RULE   2     clientid IN msclient nea clientid IN msclient
    Message:Client ID is a duplicate
RULE   3     paxclass exis
    Message:Missing passenger classification
RULE   6     putime ge 0700
    Message:Trip time is too early!
RULE   7     rt eq y
        or   rt eq n
    Message:Round trip code is incorrect
RULE   8     paxclass eq e
        or   paxclass eq eh
        or   paxclass eq h
        or   paxclass eq weh
        or   paxclass eq wh
    Message:Passenger code is incorrect
RULE   9     trippurp eq hc
        or   trippurp eq nu
        or   trippurp eq mow
        or   trippurp eq se
        or   trippurp eq ft
        or   trippurp eq gt
        or   trippurp eq adc
        or   trippurp eq dl
    Message:Trip code is incorrect
RULE  10     endhour IN vehops gta sthour IN vehops
    Message:End hours must be greater than start hou
RULE  11     endmile IN vehops gta stmile IN vehops
    Message:End miles must be greater than start mil
RULE  12     endmile IN vehops gta stmile IN vehops
    Message:End miles must be greater than start mil
RULE  13     endhour IN vehops gta sthour IN vehops
    Message:End hours must be greater than start hou
```

FIGURE 2-16. RULES FOR CAR DATA BASE

characteristics as prsched (only the data will be different). To create realtime, you can immediately enter "relations" and associate realtime with the attributes you just created for prsched. It will then be necessary to re-enter rules and passwords.

On the other hand, the master client relation (called msclient) will have some of the same attributes as the schedule relations (such as clientid and lastname) but also new attributes (such as birthdate, zip code, and sex). In this case, you would first define the new attributes (you don't need to redefine existing attributes). You would then associate the new relation msclient with the appropriate attributes (both new and previously defined).

2.2.5.8 Altering Data Base Structure

Once the data base structure has been created, it is not an easy process to change it. That is why we stress conceptualizing the data base and defining attributes on paper first. Nevertheless, changes will often be necessary. Of course, new relations can be established at any time and loaded with data from scratch. Other, more complicated restructuring procedures are described briefly below:

- Rules: Rules cannot be changed, they can only be deleted and rewritten. Note in Figure 2-16 that there are no rules numbers 4 and 5. They contained errors and were deleted. To delete a rule, enter the command displayed in Figure 2-17.

```
R> Delete rows from RBSRULES where numrule eq n
```

FIGURE 2-17. DELETE RULES COMMAND

- Keys: Keys assigned to specific attributes can be deleted and new keys can be added. These commands are displayed in Figure 2-18.

```
R> Build key for attribute name in relation name
R> Delete key for attribute name in relation name
```

FIGURE 2-18. DELETING AND ADDING KEYS COMMANDS

- Renaming Features: Relation, attribute and owner names can be changed by means of the commands displayed in Figure 2-19.

```
R> Rename attribute name to new attribute name (in relation name)
R> Rename relation relation name to new relation name
R> Rename owner owner name to new owner name
```

FIGURE 2-19. RENAMING COMMANDS

- Redefining an Attribute: In order to redefine an attribute (for example, to change the length of a text field), you must create a new relation with the corrected attribute and unload the data in the old relation to a temporary storage file while you make the change, then reload the data to the new relation. This is a cumbersome, time consuming procedure described in Section 2.4.2 of the R:base manual. This procedure is a major weakness of R:base. You should make every effort to correctly define attributes in the first place.

- Deleting Attributes from a Relation: The removal of an attribute from a relation entirely is considerably easier than altering an attribute. The project command, as shown in Figure 2-20, enables you to create a new relation containing all or only some of the attributes of an old relation. The project command plays an important role in setting up the scheduling system explained in Section 2.3.

```
R> Project new relation name from relation name using (all) +
   (attribute name 1) (attribute name 2) ----
```

FIGURE 2-20. PROJECT COMMAND

- Adding Attributes to a Relation: Attributes can be added to a relation by means of the union command. This command can also be used to combine two existing relations as will be shown in Section 2.3. Define the new attributes and associate them with a new relation. Then, use the union command to combine the existing relation and the new relation as shown in Figure 2-21.

```
R> Union relation name 1 with relation name 2 forming
   relation name 3.
```

FIGURE 2-21. UNION COMMAND

Once the combined relation is formed, you may want to remove the old relations. This process is explained below. You must add data to the new attributes. This can be done by means of the load module (see Section 2.2.6).

- Removing Relations: You may have no use for an old relation, particularly after creating a new relation through the project or union commands. To remove a relation, simply enter "remove relation

name". R:base will respond **Press [Return] to remove the relation relation name. Press [ESC] to stop the command.** This response gives you a second chance before taking the drastic step of killing an entire relation.

- Reload Data Base: When you delete elements of a data base, the space they occupied remains filled. This is particularly significant when you replace a relation through the union or project commands. Even after deleting the old relation, the disk space which it occupied remains full. This can be a major problem when using floppy disks. This problem can be resolved by reloading the entire data base (individual relations cannot be reloaded). To reload, simply enter "reload new data base name". If you are using floppy disks, you can reload to either the a or b drives, depending on which has more blank space. The second R:base diskette (which will be in drive a) has about 150K bytes available. If you reload onto the R:base diskette, you will later want to copy it back to your data disk using your operating system's copy function. If you reload to drive b, you must specify "b:" before the new data base name.

  Prior to reloading on either drive, be certain that you have enough storage space for both the old and new data bases. Appendix B discusses storage space requirements. You will need sufficient space only to accommodate the size of the new, reorganized data base which will be smaller than the original data base. If you attempt to reload without sufficient disk space, you will kill the data base. Be certain you have a back-up copy.

2.2.6 Entering and Changing Data

There are four methods of loading data into an R:base relation: (1) the load module; (2) the prompt (load) module; (3) the forms module; and (4) data from a file outside of R:base. The latter is a specialized function beyond the scope of this documentation. We will assume that you will be loading data originally into R:base. Of the three methods for doing so, forms is by far superior. Forms represents, in effect, a fifth mode to go along with prompt, help, load and define. Section 2.2.6.1 describes how to enter data, while Section 2.2.6.2 describes how to change data.

2.2.6.1 Entering Data

The focus of this section will be on entering data by means of the forms mode. In order to enter data by means of the prompt mode, enter "prompt load" and follow the directions within prompt. To enter data by the load module, enter "load" to access the load mode. Then enter each record attribute by attribute leaving a blank space between each value. At the end of each record, hit "return" and begin again on the next line. The disadvantage of this method is that you have no prompts and must keep a list of attributes at hand for reference. In a large relation with many attributes, this can obviously cause problems. You can also not return to a previous line for corrections, but must leave the load module and employ one of the change functions described in Section 2.2.6.2.

The advantage of the forms mode is that you can create a customized data entry form for each relation, thus ensuring consistent and accurate data entry by your staff. Figure 2-22 describes how to enter the forms definition mode.

```
R> forms
   Begin R:base Forms Definition
   Enter form name: Schedule
   Enter relation name: prsched
   Create or edit your form, push [ESC] when done
```

FIGURE 2-22. ENTERING FORMS DEFINITION

As is shown, each form must be named and associated with one (AND ONLY ONE) relation. This is a weakness in that we desired to use the same form to enter data in both prsched and realtime (since they had the same structure). We had to create the same form twice.

Once inside forms definition, you will be faced with a blank screen on which to design a data entry form. Use your computer's cursor control and screen editing functions to design a form. Remember, this form is for internal use only and should be designed to facilitate the ease of data entry. There are really only two constraints on your design. You must leave enough space between data items to accommodate the desired maximum length of each attribute value. Secondly, data entry will occur in the order in which you associated attributes with a relation in the define mode. You should probably set up your form in the same order. Otherwise, the cursor will jump randomly from one item to another instead of proceeding in sequence. This will slow down data entry.

Figure 2-23 displays the form called "schedule", which we created to enter data into the relation prsched. As you can see, we attempted to group related data. Trip data and vehicle # appear on the first line, separated from the client data which is grouped together following the blank line. The next group of data is related to the specific trip. This is followed by trip purpose and payment. The last line includes only edit date. Note that you should use terms which are meaningful to the data entry personnel, and not necessarily your defined attribute names. "S" and "E" indicate the starting and ending points for each field's data entry (see below).

```
Trip Date  S      E          Vehicle #   SE

Client ID S  E   Last Name  S              E      First Name  S         E
Address   S              E      City  S         E
Passenger Class  S E       Special Needs  S         E

Pick-Up Time     S         E      Destination  S              E
Round Trip?      E                Return Time  S         E
Trip #   S         E

Trip Purpose  S E      Payment    S  E

Edit Date  S      E
```

FIGURE 2-23. SAMPLE FORM

You can check on your form structure by entering "select all from forms" while in the R:base mode. In addition to the visual layout displayed in Figure 2-23, you will also receive a numeric layout description.

Once you have completed form design, hit [ESC] to receive the following menu at the top of the screen:

E(dit), L(ocate attributes), Q(uit):

This is the first of a series of menus which you will receive within forms. To chose an option, enter its first letter and press return. If you are not satisfied with the design of your form, enter "E" and and return to form design. Otherwise, enter "L". You will receive the following response in our sample case:

Current location for veh# - K(eep), S(et or change), D(elete):

Veh# is the first attribute associated with the relation prsched. You will want to locate it following "vehicle#" on the form. Enter "S" (set). You will receive the following instruction:

Current location for veh# - Move cursor to start location and press (S)

Place the cursor a few spaces after vehicle# where you would like data entry to start (hence the S). Enter "S" to mark this position. The cursor will then move automatically to the last possible end position for that piece of data as defined by you in attributes. In the case of veh#, the maximum length is "2". You will then receive the following prompt:

Current location for veh# - Move cursor to end location and push [E]

Most likely, you will want to locate E at the maximum length you have established for the attribute. You do have the option, however, of changing it. After entering "E", you will be returned to the set menu to repeat the process for each attribute in turn. When all attributes have been located, you will be returned to the main forms menu.

If you want to relocate any attributes, enter "L" and return to the locating function. You will be prompted with the name of each attribute in turn. If you do not want to change its location, enter "K" (Keep) and proceed to the next attribute. If you do want a change, enter "S" (set or change) and locate it as before. Unfortunately, you must proceed through all attributes in this fashion even if you only want to change one.

Once your form is completely as you want it, enter "Q" to quit forms and return to the main R:base module. Data entry is now a simple process. Type "enter schedule" (schedule is our form name) and the form will appear on screen. A bar shaped cursor will appear in the assigned space for our first attribute, veh#. Enter the data and hit [return]. The cursor will automatically move to the next attribute in sequence. It is impossible to enter data which exceeds the assigned length of each attribute. You may leave attribute fields blank unless you established a rule requiring an entry (the "exists" operator). After entering a single record (all attributes), you will receive the following menu:

A(dd this data), R(euse data after adding) E(dit), Q(uit)

If you enter "A", the data will be written to disk and a blank form will reappear for your use. (You may hear the computer writing to disk, although sometimes it may store several records in memory before writing. This helps to speed up the process.) If you enter "R", the data will also be written, but you will receive a form containing all the data you just entered. This is a highly desirable feature which allows you to change a small number of data items. For example, suppose you were booking the same trip for one client everyday of the month. Instead of retyping all the client data, you would simply need to reenter trip date. Move the cursor around to the attribute you want to change, leaving everything else untouched. In this case, you would only need to change the "day" in trip date. You can use your screen editor to move around within the bar cursor, changing individual characters rather than the whole entry. The use of this feature can greatly reduce the time involved in data entry.

If you realize you have made a mistake, enter "E" to edit the record before writing it to disk. When you complete the data entry session, enter "Q" to quit to the main R:base module.

If you had established rules for this relation during the definition phase, you will receive an error message should you try enter data which violates a rule. The data will not be written to disk until you correct the error. R:base itself will also spot data errors such as incorrectly formatted dates or times. For example, R:base knows that the 06 month has only 30 days, and the 02 month has only 28 except for certain years in which it has 29. This feature will spare you the embarrassment of scheduling a trip for June 31!

2.2.6.2 Changing and Deleting Data

Data can be changed in R:base by going back and editing the forms which you created, or by using the change command. The format of these two commands is shown in Figure 2-24.

```
R> Change attribute name to value (in relation name)* where ---
R> edit using form name (sorted by ---) (where ---)


*You only need to enter the relation name in commands of this
 type if the attribute is associated with more than one relation,
 but you want to change the value in only one relation.
```

FIGURE 2-24. CHANGE AND EDIT COMMANDS

The advantage of the change command is that you can pinpoint exactly which records you want to change. The disadvantage is that you have no form or prompts for reference. On the other hand, the edit command enables you to make changes right on your form. The disadvantage is that you have to sort through one record at a time.

Suppose you prescheduled one week worth of trips for a client and the client wanted to change the pick-up time (putime) from 0830 to 0930 for each day. This change could be accomplished by one change command as follows:

```
R> "change putime to 0930 in prsched where clientid eq 0140 and tripdate eq +
R>  05/01/79 or tripdate eq 05/02/79 or tripdate eq 05/03/79 ---"
```

As with all R:base conditional string commands, you are limited to a maximum of ten "where" conditions. The arithmetic operators ("eq") are the same throughout R:base as shown previously in Figure 2-10. Again, the English is a little stilted in that you have to repeat the attribute name ("tripdate") for each "where" condition. The disadvantage of not having any prompts is apparent in a lengthy entry such as our example. If you make one little typing mistake, you will receive a syntax error message and have to enter the entire sentence again.

Note the use of "and" and "or" in the "where" clause. You only want to change records which contain the clientid "0140". Thus, this condition is followed by "and". However, any one of the tripdate conditions can satisfy your requirements (you could search long and hard for a single record which contains a trip on more than one day). Thus, each tripdate condition is joined by an "or". Remember, you cannot exceed ten conditions in one change command. If you have more than ten conditions, you must initiate a second change command.

If you use the form edit command to change this example, you would have to change the pick-up time on ten separate records but first you will have to find the records by using the sort and where commands as follows:

```
R> "edit using schedule sorted by tripdate where clientid eq 0140"
```

The sort command will place the records for which you are looking first in sequence. It may not always be possible to bring the exact records you want to change to the forefront, which means you will have to flip through records you don't want to change to find the ones you do want to change.

Sorts will take place in ascending order (a,b,c ... or 1,2,3) unless otherwise specified. To sort in descending order, enter =d following the sort attribute ("tripdate =d"). Be sure to leave a blank space after the attribute name so R:base doesn't think the "=" sign is part of the name. You can sort by more than one attribute, with the first one listed taking precedence. Sorts temporarily occupy disk space (see Appendix B), so don't sort for the fun of it!

Complete records can be deleted quite easily from R:base. The two delete commands are displayed in Figure 2-25.

```
R> Delete duplicates from relation name
R> Delete rows from relation name where ---
```

FIGURE 2-25. DELETE COMMANDS

The first case is the specialized event of entering a record twice. The "delete duplicates" command will automatically recognize and delete any duplicates. The "delete rows" command will delete specific records defined by you using the "where" clause.

You should now have a fairly thorough understanding of using R:base. Sections 2.3 and 2.4 will discuss more specifically the application of these functions to paratransit management.

2.3 APPLYING R:BASE TO TRIP SCHEDULING

The trip scheduling function provides an excellent example of the power and benefits of data base management software such as R:base 4000. This section explains how to use R:base to perform this function. This process is summarized graphically in Figure 2-26.

2.3.1 Step 1: Establishing Base Relations (Files)

The first step is to establish two base relations into which you will enter trip data. In this prototype, we have named these relations "prsched" and "realtime". The techniques for defining these relations and entering data into them was described in detail in Section 2.2. These relations will have identical structures (attributes, rules and forms). The only difference between them will be in the data entered. The "prsched" relation will contain trips which are prescheduled some significant amount of time before they are to take place. The "realtime" relation will contain true dial-a-ride trips as they are called in.

Typically, demand-responsive operators will have a base of regular trips which changes little from day-to-day or even month-to-month. Mrs. Jones always goes to the nutrition site (or adult care center, dialysis facility, doctor, etc.) every Monday, Wednesday, and Friday at 11:30. It was the philosophy of our case study operator, Call-A-Ride, to schedule as many trips in this fashion as possible. In effect, a stable route structure was created with certain individuals and trips always assigned to the same vehicle. They would then "fill-in" with true dial-a-ride trips.

The advantages of this technique is in the creation of a stable, dependable, cost-effective route structure. You are essentially using the preschedule function to maximize vehicle load factors. A second major benefit of using a preschedule file will be to greatly reduce data entry requirements since most of the information for each trip will need to be entered only once, rather than 20-30 times a month. The initial data entry requirements can often be a major disincentive to automation. The disadvantage of prescheduling is the constraints in service availability placed on the occasional dial-a-ride passenger. While the balance between prescheduled and dial-a-ride trips is a local operational decision, we strongly recommend the use of a preschedule file for at least some trips.

2.3.2 Step 2: Load Prsched File

The next step involves loading data into the prsched file. This relation should be thought of as a "dummy file" which will contain all possible

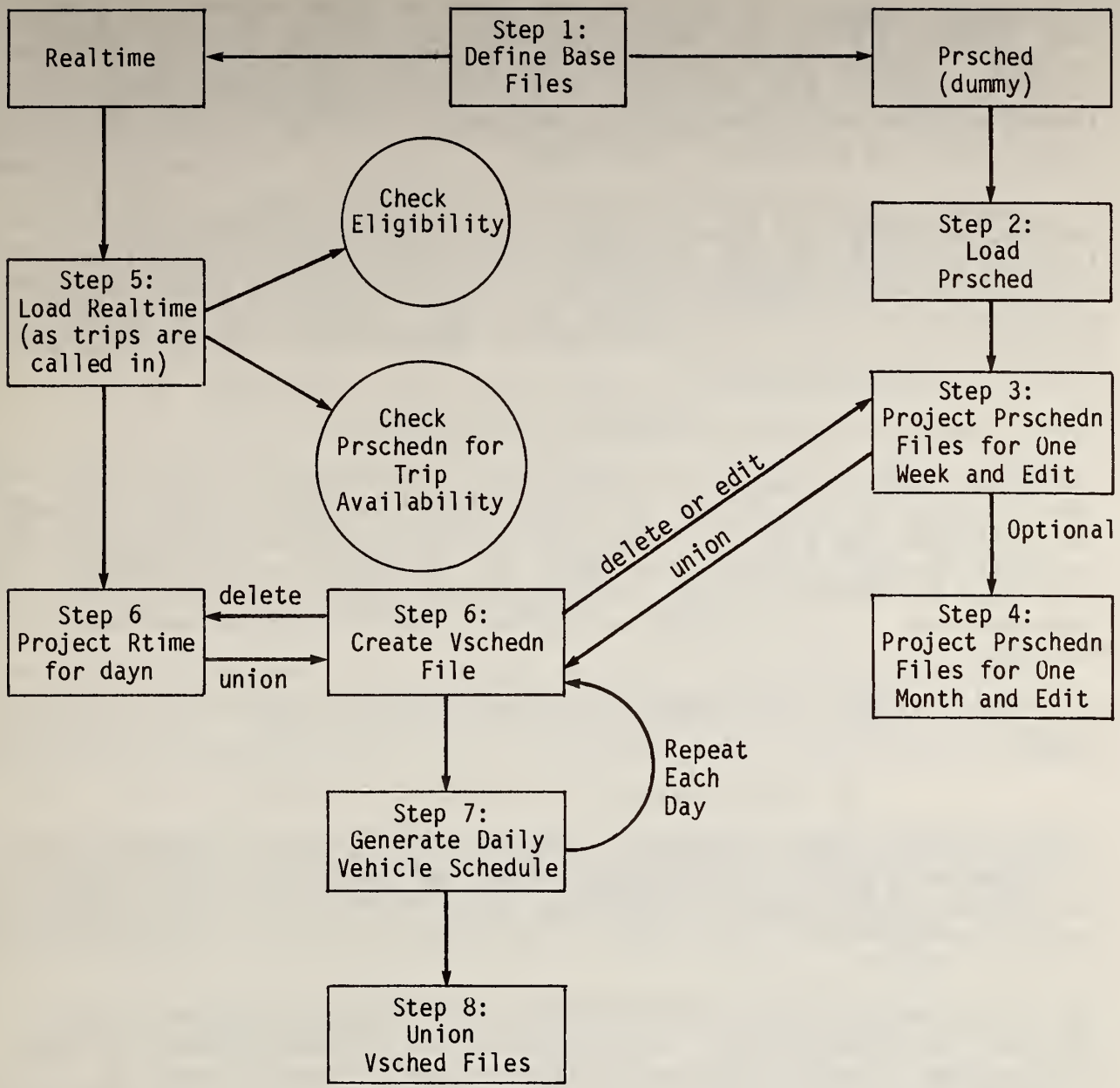


FIGURE 2-26. USING R:BASE FOR TRIP SCHEDULING

prescheduled clients. All data for each client should be entered (using the form "Schedule") as you will want it to appear on a vehicle schedule, except tripdate. Tripdate is, at this point, the true dummy feature in that it doesn't matter what date you enter. Most of this data will never change. Instead of entering it separately for each day of the month in which the client rides, you will enter it only once.

We recommend editing the dummy prsched file on a monthly basis. A month is a standard operating time frame. Accounting and reporting procedures are typically performed on a monthly basis. Client schedules will also often conform to monthly parameters, particularly if clients change travel patterns on a seasonal basis as is true of our case study. Therefore, prior to the beginning of each month, you should edit the dummy prsched file, deleting and adding clients as appropriate.

2.3.3 Step 3: Projecting Date-Specific Prsched Files

To use the dummy prsched file, you will first need to project out of it a series of date specific files. This can be done in a number of ways. If each day of the week has a fairly unique trip pattern, you may want to project five new files, one for each day of the week. Or, you may have a Monday/Wednesday/Friday and Tuesday/Thursday trip pattern which would require only two separate files. In this example, we created a separate file for each day of the week.

To project your new files, enter the following for each new file you wish to create:*

```
R> "project prschedn from prsched using all"
```

This command will cause a series of new files called prsched1, prsched2, etc., to be created out of the dummy prsched file. The new files will contain all the attributes of prsched ("using all"). In place of the "n" in "prschedn", assign a unique numeric value (i.e., 1, 2, 3, etc.) for each prsched file which you project.

You are now ready to edit each new relation. All records will need to be assigned specific trip dates. In our sample case, prsched1 represents Monday (day 1) trips. Monday's date is 05/07/79. We would therefore enter the following:**

```
R> "change tripdate to 05/07/79 in prsched1"
```

*When you project a new relation out of an existing one, no features of the original file such as forms, rules or keys are projected. Only the attributes and data are projected. Since you will not actually enter data into these files, the absence of these features is insignificant. You do have to be careful that in editing (see below) you don't violate any of your data entry rules.

**You must use the change command to accomplish this. You cannot "edit forms" because the form "schedule" is associated only with the prsched dummy file.

This will assign the correct trip date to all records in prsched1. It is important to specify the relation name (prsched1), otherwise tripdate will be changed in all relations. It is not necessary to specify any "where" conditions because you want to change all records in prsched1. Change the other prschedn files to the appropriate dates in the same way for the first week of the month.

You will next need to go back into each file and delete or change individual records. Remember, each prschedn file contains all possible prsched records housed in the dummy prsched file. But you know that many of these clients do not ride on Mondays (or M-W-F depending on how you are doing it). You need to delete these records from prsched1 as shown below:

```
R> "delete rows from prsched1 where clientid eq 0007 or +
   clientid eq 0043 or clientid eq 0101 ...."
```

Again, be certain to include the relation name ("prschedn") to avoid deleting the records from all files. You are limited to ten (10) "where" conditions per delete command. Also, make certain you join the conditional "where" string with "or's". The computer will search long and hard for a record that contains both clientid 0007 and clientid 0043. You may also want to change pick-up times or other attributes on certain records.

Due to the limitation of ten conditions in each string command, editing the prschedn files can be time consuming if your preschedule is highly variable from day-to-day. If you anticipate the need to make many changes from day-to-day in your preschedule file, we recommend adding a date code attribute to the preschedule file.

This attribute could work in several ways. For example, suppose most of your trips conformed to a M-W-F (Monday, Tuesday, Wednesday) or T-Th trip pattern. You could create a one-column attribute with data entry limited to "1" (for M-W-F trips) or "2" (for T-Th). Then, when projecting your prschedn files, simply enter:

```
R> "project prschedn from prsched using all where +
   datacode eq 1"
```

Then, you would simply need to change the trip date as before and edit the few trips which will inevitably change or be canceled from week to week. If your trip patterns are really unique to each day and/or client, you might want to create a more complex code. You could, for example, create a numeric code for every possible travel combination for the five days (i.e., M = 1, M-T = 2, M-T-W = 3, etc.). When projecting files for each day, make sure your "where" clause contains all possible codes for travel on that day.

Perform this function on each prschedn file. You have now completed prescheduling for the first week of the month. Of course, as the month proceeds, there will certainly be ad hoc changes to the preschedule. Mrs. Jones will become ill and cancel, etc. These changes can be entered into the appropriate prschedn file as the month proceeds.

2.3.4 Step 4 (Optional): Project/Edit Prschedn Files for Rest of Month

The timing for the performance of this step depends on how far in advance you want to be able to schedule real time trips. To schedule real time trips (Step 5) you need a base of prescheduled trips around which to plug in the dial-a-ride trips. You now have that base for one week. If that is sufficient, you should skip this step and proceed to Step 5.

If, however, you want to schedule dial-a-ride trips further in advance, you must project additional prschedn files out of the existing ones. For example, suppose we want to have a prescheduling base for the entire month and we are using an individual file for each day of the week. After you have edited prsched1, the Monday (or day 1) file, you must project a new file out of it for the following Monday. You will then edit this file for the correct tripdate. You will repeat this process two more times for the remaining two Mondays in the month. You will then repeat the process for each day of the week until you have 20 or so active prschedn files, one for each service day of the month. Projecting a new file requires a single command and takes little time. The alternative is to enter each prescheduled trip every time it is scheduled during the month, an extremely time-consuming process.

This method has operational advantages but R:base disadvantages. The further in advance you schedule dial-a-ride trips, the more they become almost like prescheduled trips. This will enhance the efficiency of your operation. The R:base disadvantage is that instead of having five active prsched files, you have twenty. That represents a substantial increase in storage requirements.

2.3.5 Step 5: Loading Real Time Trips

There are two factors involved in loading real time trips. These are checking for vehicle availability and client eligibility. Trip requests can be checked against the prschedn files to determine vehicle availability for a specific date. The more you have organized your service pattern into a series of informal routings, the easier it will be to check on vehicle availability since you will know what vehicle to look at for a specific trip type or area. You can access a specific vehicle schedule by means of the "select" command as follows:

```
R> "Select all from prschedn sorted by putime where veh# eq A1"
```

This will display the prescheduled trips for vehicle A1 on the date of the prschdn file. Since you can only view 132 columns at a time, it is important that if your schedule relations exceed 132 columns, trip schedule information such as pick-up time, location, destination, and round-trip appear in the first 132 columns*. Otherwise, you will have to list each attribute you want to view instead of typing "select all". This would be uncomfortably time consuming with a live client on the line.

*Even though your screen may only display 80 columns on a single line, the remaining columns will be "wrapped around" into a second line immediately below the original line.

Even before checking for service availability, you will want to determine client eligibility. Client eligibility can be checked by means of the master client file which will be the first relation you will create in designing your data base. You will want to check on client eligibility each time a real time trip request is called in.

Figure 2-27 displays a report generated from the master client relation called "msclient". The process of report generation will be described in Section 2.4. The complete client file for the case study data base is displayed in Appendix D.

This report contains all the information needed for a dispatcher (or trip scheduler) to perform two functions: 1) book a specific trip into the realtime file, and 2) check client eligibility.

Each dispatcher should have a hard-copy client file for reference at their work station. In many cases, we think this is actually faster than looking up the client record in the computer and displaying it on screen. Nevertheless, if you prefer to operate in the latter fashion, a specific client record can be called up by using the "select" command as follows:

```
R> "Select all from msclient where clientid eq 0032"
```

Client records can also be viewed even more efficiently by means of the customizing features described in Appendix C.

You will most likely chose to select client records by client ID or last name. We recommend client ID since it is guaranteed to be unique. Again, you must be sure that the attributes you need to see are within the first 132 columns, or use the customizing features described in Appendix C.

As can be seen from the client file report, we have requested (when printing the report) that it be sorted alphabetically by last name. Alternately, you might want to sort by client ID. However, many clients don't know or remember their ID numbers so we tend to favor an alphabetical sort by last name for this purpose.

The information from last name through special needs (specneeds) will enable you to book a trip in "realtime" without asking the client anything except destination and trip purpose. The remaining items will enable the dispatcher to check eligibility. These items include specific program ID number (in this case medicaid and Title XX); Title XX available units; a termination date; and a termination reason. As you can see in Figure 2-27, several clients are no longer eligible.

Once you have checked on client eligibility and the availability of service, you can then enter the trip into "realtime" using the form "schedule" created for that purpose.

MASTER CLIENT FILE

Last Name	First Name	ID	Address	City	class	SpecNeed	Med#	XX#	units	Term Date	Term Why?	EditDate
Abbott	Catherine	0064	9702 Bridge	Yarmouth	e	no phone				05/31/79	moved	05/31/79
Allen	Helen	0146	50 Lawtner	Centervill	eh		628982					05/06/79
Allenson	Herbert	0154	55 Rustown	Cotuit	weh			251753	30	09/30/79		05/14/79
Aries	Rose	0077	15 Fringewood	Dennis	e	medicatio						05/03/79
Armas	Marie	0071	12 Snow	Dennis	e							05/16/79
Auld	Grace	0031	27 Tealford	Dennis	eh		019952	275743	22	08/31/79		05/15/79
Auld	Nellie	0149	108 San Marcus	Falmouth	eh		043904	956433	34	09/30/79		05/11/79
Avallon	Kathleen	0072	1 Vickie	Dennis	e	structure						05/17/79
Baker	John	0001	65 Bass	Dennis	e							05/01/79
Barrett	Robert	0129	66 Kenwood	Falmouth	e		293173					05/11/79
Battis	Edith	0155	481 Oates	Hyannis	weh		986152					05/15/79
Bensten	Ellen	0090	4 Beachway	Hyannis	eh	structure						05/12/79
Berman	Helen	0006	23 Sheryl	Yarmouth	e	medicatio						06/11/79
Biden	Manuel	0150	29 Coronation	Falmouth	eh							05/12/79
Boggs	Vickie	0122	13 Dyer	Pocasset	eh		470145					05/17/79
Boisvert	Mildred	0056	3 Alford	Dennis	e							05/09/79
Boren	Hilda	0148	10 Downing	Falmouth	e					05/31/79	overdue bill	05/31/79
Boyd	Thomas	0160	55 Astor	Sandwich	eh							05/19/79
Bradley	Ruth	0041	5 Ryan	Yarmouth	eh							05/23/79
Buckley	Marguerite	0091	108 Syracuse	Falmouth	e	escort						05/13/79
Buckner	Marion	0042	196 Sandra	Yarmouth	h							05/24/79
Buckner	Eliza	0021	109 Lance	Hyannis	eh		017220	111341	12			05/09/79
Bumpers	Blanche	0143	120 Ocean Ave	Falmouth	eh		022116					05/05/79
Busheuff	Mary	0157	4 Point East	Bourne	eh	medicatio	920527					05/17/79
Cabrera	Florence	0162	73 Larry	Bourne	weh		695160					05/21/79
Callahan	Arthur	0049	608 Mercedes	Yarmouth	e							05/04/79
Carmichael	Lillian	0027	19 Fairhaven	Yarmouth	e							05/12/79
Chaffee	Claire	0045	9 Woodwind	Dennis	e		167177	497833	02	05/31/79		05/30/79
Chiles	Bertha	0082	25 Eastwood	Harwich	e							05/05/79
Clark	Paula	0118	1578 Samuell	Falmouth	h		144202					05/20/79
Clear	Alice	0164	89 Eastwood	Mashpee	eh							05/31/79
Clemens	Vanessa	0063	43 Harper	Yarmouth	e			176443	50	12/31/79		05/12/79
Cohen	Dora	0019	38 Lucille	Yarmouth	e							05/07/79
Considine	Gladys	0026	15 Healy	Yarmouth	h		270766			05/31/79	deceased	05/31/79
Craig	Helen	0133	79 Sidney	Falmouth	eh	medicatio	876847	277433	43	01/31/80		05/08/79
Craig	Alice	0017	504 University	Harwich	e							05/05/79
Crawford	Alida	0103	5 Little Pocket	Yarmouth	e							05/21/79
Curren	Cindy	0142	62 Parkhurst	Falmouth	h		509556					05/03/79
Cutler	Miriam	0032	21 Woodale	Yarmouth	e			415443	0	04/30/79		05/16/79
Davis	Evelyn	0102	201 Bella Vista	Dennis	e							05/20/79
Denny	Charles	0055	79 Patty	Yarmouth	e							05/08/79

FIGURE 2-27. MASTER CLIENT FILE

Should a client's status change as a result of the telephone conversation, the dispatcher should immediately update the client record. For example, suppose Downey's trip request exhausts her 2 remaining units of Title XX eligibility. After booking the trip, the dispatcher would enter the following:

```
R> "change XXunits to 0 in msclient where clientid eq 0097"
R> "change editdate to 05/16/79 in msclient where clientid eq 0097"
```

Other information regarding the client such as address, classification (someone could cross the elderly threshold), last name (change in marital status), or special needs might also require updating. This can be accomplished by means of the "change" command as shown above or by using the customizing features described in Appendix C. Whenever a client file is updated, it is important to change the edit date as well in order to provide an audit trail.

Updating fields in the client master file demonstrates the true power of a relational data base. For example, suppose that you have already prescheduled a number of trips for a client who then moves (a not unlikely occurrence). You will not want to go back into each vehicle schedule file and change the address. Since the address field is shared by almost all relations in our data base, they will all be updated automatically when you make the initial change in the master client file. Only a data base with relational features will function in this manner.

An alternative method of scheduling dial-a-ride trips would be to dispense with the realtime file completely and load these trips directly into the prschedn file of the appropriate date. While this method has some advantages in simplifying the process, it has one major disadvantage. To schedule trips into prschedn files directly, you would need to create a data entry form for each prschedn file or load data without a customized form. If you want to book trips a month in advance, you would have to create 20 or so forms, a time-consuming task. Remember, each form is associated with one and only one relation. We consider loading without a customized form to be a serious problem.

2.3.6 Step 6: Combining Prschedn and Realtime Relations

You are now ready to begin combining prescheduled and real time trips which have been scheduled prior to day 1 of the month. This process should be initiated at the cut-off point for the acceptance of real time reservations. This is usually 24-hours in advance of the trip date.

Project out of your "realtime" file a file containing only trips for day 1. Use the following command:

```
R> "Project rtime from realtime using all where tripdate eq 05/07/79"
```

Next, union rtime with the prschedn file for day 1 as follows:

```
R> "Union prsched1 with rtime forming vsched1"
```

The "vschedn" file will be the final repository of all trip information for the month. In the above command, you do not need to specify attributes. If none are specified, "all" is assumed by R:base.

You now have a file ("vschedn") containing all trips (both prescheduled or real time) for the first day of the month (or 05/07/79 in our sample data). You will have no further use for the "rtime" file since these trips now reside in "vschedn". Remove this relation as follows:

```
R> "Remove rtime"
```

You will also, at this point, have no further use for the prsched1 file since its trips also reside in the "vschedn" file. If you performed step 4 and projected new file(s) out of prsched1, you can now delete it by entering "remove prsched1". If, however, you are working one week at a time, you will now want to prepare "prsched1" for use on the next appropriate day. In our sample, that day would be Monday, 05/14/79, since we assume that trip patterns for each day of the week are substantially the same from week to week. Change the records in prsched1 as follows:

```
R> "change tripdate to 05/14/79 in prsched1"
```

Change individual records as needed.

You will follow the same procedure on each subsequent day. For example, on day 2 you will "union" prsched2 with the new rtime file for day 2 forming vsched2.

2.3.7 Step 7: Generating Vehicle Schedules

Once you have combined all trips for a specific day in the vschedn file, you will want to produce vehicle schedules for use by drivers and dispatchers. Figure 2-28 displays an example of a vehicle schedule. As you can see, this has been sorted by pick-up time and includes all information necessary for drivers and dispatchers to perform their jobs.

The next section, 2.4, describes in detail the process of report generation. After performing the functions described in that section, this report could be generated with the following command:

```
R> "Print reportname sorted by putime where veh# eq A1+
R> and tripdate eq 05/07/79"
```

The key point is that you want to print a separate report for each vehicle for each day to tear off and hand to the appropriate driver. This is accomplished through a series of "where" conditions.

2.3.8 Step 8: Combining Daily Schedules

As the month proceeds, you will be creating a vschedn file for each day containing all the prescheduled and real time trips for that day. At some point in the process, you will need to combine these daily schedule files so that you end up with a single trip file at the end of each month. This file

VEHICLE SCHEDULE
Vehicle #: A1
Date: 05/07/79

Time	Last Name	First Name	Address	City	Destination	RT Time	Class	Purpose	Special Needs
700	Allen	Helen	50 Lawtner	Centervlle	Dialysis		eh	dl	
700	Sandman	Theresa	1507 Vinewood	Hyannis	Dialysis		n	dl	structure
900	Tower	Nicholas	27 Breaker	Pocasset	BCH	1500	weh	adc	
900	Craig	Helen	79 Sidney	Falmouth	BCH	1500	eh	adc	medication
900	Sherman	Milton	1 Cold Ridge	Bourne	BCH	1500	e	adc	medication
900	Barrett	Robert	66 Kenwood	Falmouth	BCH	1500	e	adc	
915	Lapon	Peter	1501 Beacon	Falmouth	BCH	1500	eh	adc	
915	Pryor	Avis	10 Pilgrim	Pocasset	BCH	1500	eh	adc	
930	Luger	John	Fishfry	Falmouth	BCH	1500	wh	adc	
930	Mathias	Milton	29 N	Pocasset	BCH	1500	wh	adc	
945	Cabrera	Florence	73 Larry	Bourne	BCH	1500	weh	adc	
945	Seal	Marie	70 Moreland	Centervlle	BCH	1500	eh	adc	
1000	Biden	Manuel	29 Coronation	Falmouth	BCH	1100	eh	nc	
1000	Curren	Cindy	62 Parkhurst	Falmouth	Dr Halgarth	1100	n	nc	
1000	Cabrera	Florence	73 Larry	Bourne	Fal Hospital	1100	weh	nc	
1130	Griffin	Genaviev	4 Poppy	Falmouth	Fal Hospital	1230	e	hc	
1200	Stanley	Harriet	191 Hill Glen	Falmouth	Fal Hospital	1300	weh	hc	

FIGURE 2-28. SAMPLE VEHICLE SCHEDULE

will serve as the basis for most of the end-of-the-month reports generated in Section 2.4. These daily files can be combined in one session at the end of the month or periodically during the month. We recommend combining these files on a daily basis beginning on day 2 when you will first have two files. Again, you will need to use the union command to combine two relations to form a third. You can only union two relations in one command in R:base as follows:

R> "union vsched1 with vsched2 forming vsched3"

After completing this union, you will have no further use for the old vschedn files. They should be deleted using the "remove relation name" command.

If you use floppy disks, you will want to consider "reloading" the data base each time you delete records or relations. This reduces the overall size of the data base. However, in order to reload, you will need space on the disk equal to the size of the original data base minus whatever you deleted. This issue is discussed more fully in Appendix B.

This process illustrates the obvious limitations of the floppy disk approach. Each time you project new files or union to create a third file, you tremendously decrease available disk space. We were able to perform this function for a one-week period in a five vehicle system. We may have had enough room for two weeks.

2.4 REPORT GENERATION

The ability to generate reports, both the print-out of a vehicle schedule shown in Figure 2-28 and end of the month statistical summaries, is a second powerful feature of R:base. Figures 2-29, 2-30, and 2-31 display three examples of "end-of-the-month" reports which can be generated by R:base.

The report displayed in Figure 2-29 is part of a monthly statistical analysis of ridership by passenger classification. This report indicates that 48 trips were accounted for during the week of May 7-11, 1979, by passengers classified as "weh" (wheelchair, elderly handicapped). Figure 2-30 displays a bill accounting for all medicaid trips during the same time period. This report also computes the total cost of these trips. Figure 2-31 displays a report which compiles various performance measurements. Because it uses the most report features, we will use this report to explain the report generation process in Section 2.4.1. In Section 2.4.2, we will discuss some other aspects of report generation demonstrated in Figures 2-28, 2-29, and 2-30. All report formats are contained in Appendix D.

2.4.1 Measuring Performance through an R:base Report

2.4.1.1 Context

Report generation should ideally take place at the end of the month. When you have completed a monthly cycle of scheduling as described in Section 2.3, you should generate reports for the month. If you are using floppy disks, you may need to generate reports more frequently so that you can remove the data and clear disk space.

RIDERSHIP BY CLIENT

ID	Last Name	First Name	Address	City	Class	Purpose	Pay	Trip Date	Trip#	Edit Date
0012	Guttierez	Isabelle	14 Marigold	Yarmouth	weh	hc	bill	05/08/79	2	04/30/79
0012	Guttierez	Isabelle	14 Marigold	Yarmouth	weh	hc	bill	05/10/79	2	04/30/79
0012	Guttierez	Isabelle	14 Marigold	Yarmouth	weh	hc	bill	05/11/79	2	04/30/79
0016	Locke	Cathleen	17 Fringewood	Yarmouth	weh	hc	bill	05/11/79	1	05/09/79
0054	Theobold	Roger	134 Forest	Hyannis	weh	hc	fare	05/10/79	1	05/08/79
0105	Stapleton	Elizabeth	29 Olson	Dennis	weh	nu	nu	05/08/79	2	04/30/79
0105	Stapleton	Elizabeth	29 Olson	Dennis	weh	nu	nu	05/09/79	2	04/30/79
0105	Stapleton	Elizabeth	29 Olson	Dennis	weh	nu	nu	05/11/79	2	04/30/79
0134	Tower	Nicholas	27 Breaker	Pocasset	weh	adc	adc	05/07/79	2	04/30/79
0134	Tower	Nicholas	27 Breaker	Pocasset	weh	adc	adc	05/09/79	2	04/30/79
0134	Tower	Nicholas	27 Breaker	Pocasset	weh	adc	adc	05/10/79	2	04/30/79
0139	Stanley	Harriet	191 Hill Glen	Falmouth	weh	hc	bill	05/07/79	2	04/30/79
0139	Stanley	Harriet	191 Hill Glen	Falmouth	weh	hc	bill	05/08/79	2	04/30/79
0139	Stanley	Harriet	191 Hill Glen	Falmouth	weh	hc	bill	05/09/79	2	04/30/79
0139	Stanley	Harriet	191 Hill Glen	Falmouth	weh	hc	bill	05/10/79	2	04/30/79
0139	Stanley	Harriet	191 Hill Glen	Falmouth	weh	hc	bill	05/11/79	2	04/30/79
0154	Allenson	Herbert	55 Rustown	Cotuit	weh	hc	xx	05/08/79	2	04/30/79
0154	Allenson	Herbert	55 Rustown	Cotuit	weh	hc	xx	05/10/79	2	04/30/79
0162	Cabrera	Florence	73 Larry	Bourne	weh	hc	med	05/07/79	2	04/30/79
0162	Cabrera	Florence	73 Larry	Bourne	weh	adc	adc	05/07/79	2	05/05/79
0162	Cabrera	Florence	73 Larry	Bourne	weh	hc	med	05/08/79	2	04/30/79
0162	Cabrera	Florence	73 Larry	Bourne	weh	hc	med	05/09/79	2	04/30/79
0162	Cabrera	Florence	73 Larry	Bourne	weh	hc	med	05/10/79	2	04/30/79
0162	Cabrera	Florence	73 Larry	Bourne	weh	hc	med	05/11/79	2	04/30/79
0165	Opie	Catherine	21 Cabot	Cotuit	weh	hc	bill	05/08/79	2	05/03/79

TOTAL TRIPS: 48

FIGURE 2-29. RIDERSHIP REPORT ("TRIPS")

MEDICAID INVOICE
Month: May

Med#	Last Name	First Name	Trip Date	Destination	Trip#	Pay
019952	Auld	Grace	05/08/79	CCH	2	med
019952	Auld	Grace	05/10/79	CCH	2	med
043904	Auld	Nellie	05/08/79	Fal Hospital	2	med
070200	Remy	Marie	05/11/79	CCH	2	med
070980	Donadio	Ethel	05/09/79	Pondville Hosp	2	med
070980	Donadio	Ethel	05/07/79	Pondville Hosp	2	med
070980	Donadio	Ethel	05/10/79	Pondville Hosp	2	med
070980	Donadio	Ethel	05/11/79	Pondville Hosp	2	med
120922	Peeke	Mary	05/07/79	Dr Berry	1	med
120922	Peeke	Mary	05/08/79	Dr Berry	1	med
167177	Chaffee	Claire	05/08/79	Dr Maloney	2	med
398790	Sandman	Theresa	05/08/79	Dialysis	1	med
398790	Sandman	Theresa	05/09/79	Dialysis	1	med
398790	Sandman	Theresa	05/07/79	Dialysis	1	med
398790	Sandman	Theresa	05/11/79	Dialysis	1	med
398790	Sandman	Theresa	05/10/79	Dialysis	1	med
509556	Curren	Cindy	05/11/79	Dr Halgarth	2	med
509556	Curren	Cindy	05/10/79	Dr Halgarth	2	med
509556	Curren	Cindy	05/07/79	Dr Halgarth	2	med
509556	Curren	Cindy	05/08/79	Dr Halgarth	2	med
509556	Curren	Cindy	05/09/79	Dr Halgarth	2	med
541275	Rice	Aaron	05/09/79	Brewster Manor	1	med
543987	Warner	Nellie	05/09/79	Dr Copp	2	med
628982	Allen	Helen	05/09/79	Dialysis	1	med
628982	Allen	Helen	05/11/79	Dialysis	1	med
628982	Allen	Helen	05/07/79	Dialysis	1	med
628982	Allen	Helen	05/10/79	Dialysis	1	med
628982	Allen	Helen	05/08/79	Dialysis	1	med
695160	Cabrera	Florence	05/09/79	Fal Hospital	2	med
695160	Cabrera	Florence	05/10/79	Fal Hospital	2	med
695160	Cabrera	Florence	05/08/79	Fal Hospital	2	med
695160	Cabrera	Florence	05/11/79	Fal Hospital	2	med
695160	Cabrera	Florence	05/07/79	Fal Hospital	2	med

TOTAL TRIPS: 53

COST: 463.750

FIGURE 2-30. INVOICE REPORT ("MEDBILL")

VEHICLE STATISTICS

Veh#	Trip Date	Hours	Miles	Fuel	Cost	MPG	Oil	Cost	Repair	Cost
A1	05/07/79	10.0	249.0	10.9	8.500	22.84	2	2.16	radiator	45.00
A1	05/08/79	10.0	125.0	15.8	11.56	7.911				
A1	05/09/79	10.0	136.0	9.00	8.000	15.11				
A1	05/10/79	10.0	141.0	9.00	8.000	15.67				
A1	05/11/79	10.0	139.0	9.00	7.500	15.44				
A2	05/07/79	8.50	299.0	20.0	16.00	14.95	1	1.25		
A2	05/08/79	8.50	202.0	20.9	16.30	9.665				
A2	05/09/79	8.75	272.0	11.0	8.880	24.73	1	1.25		
A2	05/10/79	8.25	200.0	20.2	15.75	9.901				
A2	05/11/79	8.50	300.0	17.5	14.15	17.14				
A3	05/07/79	7.25	138.0	21.3	17.00	6.479			windshield wipe	23.75
A3	05/08/79	7.25	106.0	17.5	14.00	6.057			inspection	2.000
A3	05/09/79	8.50	227.0	21.7	17.50	10.46				
A3	05/10/79	7.25	152.0	18.5	15.00	8.216				
A3	05/11/79	8.75	139.0	14.5	12.35	9.586				
B4	05/07/79	6.00	98.00	10.1	8.250	9.703			replace tire	100.0
B4	05/08/79	6.00	97.00	10.1	8.250	9.604				
B4	05/09/79	8.00	111.0	7.00	6.000	15.86				
B4	05/10/79	6.00	74.00	8.00	7.000	9.250				
B4	05/11/79	6.50	104.0	8.00	7.000	13.00				
C2	05/07/79	8.00	300.0	11.4	9.250	26.32				
C2	05/08/79	7.00	137.0	12.4	10.10	11.05				
C2	05/09/79	8.75	301.0	17.5	14.25	17.20	2	2.94		
C2	05/10/79	8.75	128.0	11.0	9.000	11.64				
C2	05/11/79	7.25	150.0	7.40	6.000	20.27				
Total:		204.	4325.00		275.590	338.0				170.75

Average MPG: 13.5

Miles/Trip: 9.010

Trips/Mile: 0.111

Trips/Hour: 2.356

FIGURE 2-31. VEHICLE STATISTICS REPORT ("VEHSTATS")

2.4.1.2 Defining a Report

The report definition process is very similar to the forms definition process, with some added capabilities. Like forms definition, report definition is a unique R:base mode, the sixth which has been defined. You enter the reports module in the same way you enter the forms module. These commands are shown in Figure 2-32.

```
R> Reports
   Begin R:base reports definition
   Enter report name:     vehstats
   Enter relation name:   vehops
   E(dit report), L(ocate), M(ark), D(efine), S(et), H(elp), Q(uit)
```

FIGURE 2-32. ENTERING THE REPORT MODULE

You are now faced with the main reports module menu. Figure 2-33 displays the characteristics of the relation vehops (vehicle operations), for which the report "vehstats" (vehicle statistics) will be generated. Figure 2-34 outlines the report generation process in the order in which we recommend proceeding.

Name	Attribute	Type	Length	Key
Vehicle #	Veh#	text	2	
Operations Date	Opsdate	date		
Driver #	driver#	text	2	
End Mile	endmile	integer		
Start Mile	stmile	integer		
Operation Miles	opsmile	real		
End Hour	endhour	integer		
Start Hour	sthour	integer		
Operation Hours	opshour	real		
Fuel Quantity	fuelqty	real		
Fuel Cost	fuelcost	real		
MPG	mpg	real		
Oil Quantity	oilqty	integer		
Oil Cost	oilcost	real		
Repair	repair	text	15	
Repair Cost	repcost	real		
Edit Date	editdate	date		

FIGURE 2-33. VEHOPS RELATION

This relation contains a feature not previously discussed. That feature is the "assign" command. Several of the attributes in vehops represent the result of performing an arithmetic function on two other attributes. For example, opsmile (operation miles) equals endmile minus (-) st(art) mile. Opshour works in the same way. These values can be entered directly by

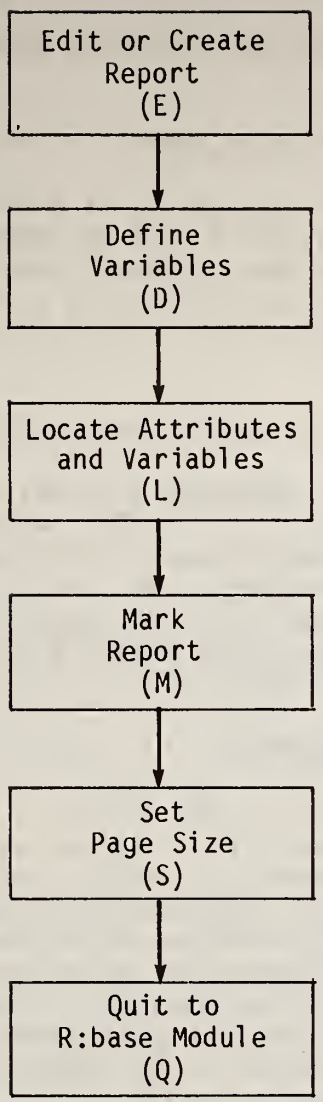


FIGURE 2-34. REPORT GENERATION PROCESS

personnel or R:base can compute them. In the case of opshour, we chose direct entry since we made all time attributes "real" instead of "time". It was therefore necessary to convert time values (0830 hours) to real numbers (8.5 hours).

In the case of opsmile, however, we instructed R:base to do the computing. This can be done as follows:

R> "Assign opsmile to endmile - stmile in vehops"

Thus, opsmile will equal end mile minus start mile. Incidentally, anticipating this function, we made a rule in vehops that end mile must be greater than start mile. Note that you must leave a space on either side of the arithmetic operator or R:base will read it as part of the attribute and tell you that no such attribute exists. We instructed R:base to compute "mpg" in a similar fashion:

R> "Assign mpg to opsmile/fuelqty"

The assign command can add (+), subtract (-), multiply (x), divide (/), and calculate percentages (%). You can not only compare one attribute (endmile) to another (stmile) as above, but compare attributes to a specific value or two values to each other. For example, we could have assigned opsmile to "endmile - 100" or to "250 - 100". If you change any attribute characteristics, make certain you also change any "assign" command to reflect these changes.

2.4.1.3 Editing/Creating a Report

You will design a report exactly as you defined a form. Keep in mind, however, that a report is intended for external uses and not internal data entry. This difference may impact your design choices. Figure 2-35 displays the design layout for the report "vehstats". "S" and "E" define the data entry areas as on forms. The letters in the left-most column delineate headings, detail and footing (see below). You can check your report design layout by entering "select all from reporter". You must select "all". This design layout is organized in the same way as the forms design layout, providing not only a visual picture of the layout (as shown on Figure 2-35), but a numeric layout description as well.

```
H                                    VEHICLE STATISTICS
H
H  Veh#  Trip Date   Hours    Miles    Fuel    Cost     MPG    Oil   Cost    Repair      Cost
H  ----  ---------  -------  -------  ------  -------  -----  ----  ------  ----------  ------
D  SE    S     E    S   E    S   E    S   E   S   E    S   E  E     S   E   S       E   S   E
F
F
F
F
F  Total:          S   E    S   E        S   E   S   E
F
F  Average MPG:                                  S   E
F
F  Miles/Trip:             S   E
F
F  Trips/Mile:             S   E
F
F  Trips/Hour:     S   E
```

FIGURE 2-35. LAYOUT FOR REPORT "VEHSTATS"

Because a report is for external use, you will probably want to pay a little more attention to the design process than was the case in designing forms. We recommend laying out your design manually first as shown in Figure 2-36 for the report "vehstats". We have provided blank layout forms for your use in Appendix E. You want to be sure to leave enough room for the maximum length of your data fields and not exceed a total width of 132 columns. If at any time you lose track of the attributes in your report, press "F3" for a display. Hit [ESC] to return to report generation.

2.4.1.4 Defining Variables

In addition to locating attributes as was done in forms, reports also enables you to define new variables and calculate their values. Variables defined in the "vehstats" report shown in Figures 2-31 and 2-36 include sums for hours, miles, fuel cost, mpg, and repair costs; and calculation of average mpg, miles/trip, trips/mile, and trips/hour. You are limited to ten variables on any one report, which is why we did not sum fuel consumption or oil cost.

Summing across in R:base is very cumbersome because you can only sum 2 attributes/values at a time. Thus, if we wanted to sum all the cost factors in Figure 2-31 across one record, we would have to sum the first two, obtain a total, add it to another, etc. Each step would use up one of our ten permitted variables. Given these limitations, it is much more efficient to transfer this data to a spreadsheet (see Section 2.5).

FIELD NAME	LINE NUMBER	ATTRIBUTE OR VARIALBE NAME	DATA LINE NOS.	BEGIN NAME COL.	END NAME COL.	NAME WIDTH	BEGIN DATA COL.	END DATA COL.	DATA WIDTH	REPORT COMPUTATION
Vehicle Statistics	3	---	--	46	63	18				
Veh #	5	veh#	7	4	7	4	5	6	2	
Tripdate	5	opsdate	7	11	19	9	11	18	8	
Hours	5	opshour	7	23	27	5	23	27	5	
Miles	5	opsmile	7	32	36	5	31	36	6	
Fuel	5	fuelqty	7	41	44	4	41	45	5	
Cost	5	fuelcost	7	50	53	4	50	55	6	
MPG	5	mpg	7	60	62	3	60	64	4	
Oil	5	oilqty	7	69	71	3	70	70	1	
Cost	5	oilcost	7	77	80	4	77	81	5	
Repair	5	repair	7	87	92	6	87	101	15	
Cost	5	repcost	7	107	110	4	107	111	5	
TOTAL:	33	--	--	5	10	6				
		sumhours	33				23	27	5	sum of opshour
		summiles	33				31	36	6	sum of opsmile
		sumflcst	33				50	56	7	sum of fuel cost
		summpg	33				60	64	5	sum of mpg
		sumrep	33				76	81	6	sum of repcost
Average MPG:	35	avempg	35	5	16	12	31	35	5	summpg/count
Miles/Trip:	37	miletrip	37	5	16	12	31	36	6	summiles/480
Trips/Mile:	39	tripmile	39	5	15	11	31	36	6	480/summiles
Trips/Hour:	41	triphour	41	5	15	11	23	27	5	480/sumhours

FIGURE 2-36. MANUALLY LAYING OUT THE REPORT DESIGN

To access the define function, enter "D". You will be presented with the following choices: **D(efine),R(edefine),Q(uit).** Enter "D" again to define new variables for the first time. You will receive the following response: **Expression.** The "expression" will consist of the variable name equal to a combination of attributes and/or values in an arithmetic expression. Setting variable expressions is done exactly the same way as "assigning" computational values to attributes. Variable names, like all R:base names, cannot exceed 8 characters. Be certain to leave blank spaces on either side of the arithmetic operator.

We defined the variables in the Vehicle Statistics report as shown in Figure 3-37. A listing of variables can be obtained at anytime while within the report mode by pressing the "F3" key. Press [ESC] to return to reports definition.

```
summpg   = sum of mpg
sumflcst = sum of fuel cost
summiles = sum of opsmile
sumhours = sum of opshour
sumrep   = sum of repcost
count    = count +1
avempg   = summpg / count
miletrip = summiles / 480
tripmile = 480 / summiles
triphour = 480 / sumhours
```

FIGURE 2-37. VARIABLES FOR "VEHSTATS" REPORT

The format of the sum variables is self-evident. The "count" variable is necessary to calculate average mpg. Average mpg equals the sum of all mpgs divided by the number of occurrences. In the vehops relation (unlike the schedule relations which contained the attribute "trip#"), we did not provide any mechanism for counting occurrences. Therefore, the count function must be established during variable definition.

The final three performance measurements in Figure 2-37 demonstrate the interaction of an attribute and a value. "480" is the total number of trips taken during the specific time period. We generated this number by means of the ridership report ("trips") shown in Figure 2-29. If we don't specify any "where" conditions, we will receive a report of all trips taken and a summation of the total number. We have then taken that number, "480" and entered it into the definition of the "vehstats" report. This value would, of course, need to be changed each month. Instructions for revising variable expressions appear in Section 2.4.2.3 below.

2.4.1.5 Locating Attributes and Variables

After defining your variables (if any), you will proceed to locate both attributes and variables on the report form. This is accomplished exactly as it was in forms. In addition to being prompted with the name of each attribute, you will also be prompted with the name of each variable which you defined.

We caution you on the location of real values (numbers with decimals). It is tricky to obtain the exact number of decimal places which you want to see on the report. It requires you to have a good idea of how many places will be occupied by the whole number. For example, you will naturally want a monetary value to have two decimal places. If you anticipate that the whole number will occupy two places, you should assign five columns from the start to the end of the value. This will work fine if you are right and the number is 99.74 (5 places including the decimal point). If, however, the number actually exceeds one hundred, it will read 101.7.

2.4.1.6 Marking the Report Layout

Each line of a report can be designated as either "heading" "detail" or "footing". Upon entering the "M(ark)" function, you will find the cursor in the left-most column. Use your cursor controls to move up and down along this column to mark lines.

Heading and footing lines contain information which you will want to appear on every page of a multi-page report. Not surprisingly, the heading is at the top and the footing is at the bottom. In the vehicle statistics report, the title, headings, and underlining should be designated with an "H" for heading. In addition, if you want a blank line between the title and headings, you must also designate that line with an "H". The footing lines begin at "total" and go to the bottom. If you want a blank line between the data and the footing, you should designate the line above "total" with an "F" as well as all the other footing lines.

Detail lines contain only data. In a report of this type, you need only designate one line as (D)etail. If you designated a second line as detail, your report would have a blank line between each line of detail. In some cases, you might want to do that, depending on the amount of data you anticipate. Do not try to create a blank line between the detail and the footing by adding an extra "D" line. Rather, you should add an extra "F" line.

Once you have marked a line, you can't immediately change it. We found this quite annoying. You must leave the mark function and then come back to it, whereupon all your marks will have been erased and you can start again.

2.4.1.7 Page Size

The default report page length is 58 lines. To alter it, enter "S" and then the number of lines you desire in response to the prompt:

Current number of lines is 58: New Number:

2.4.1.8 Printing a Report

The command to print a report is "print reportname". You may attach the usual sort and where clauses. In our example, the command will read:

R> "Print vehstats sorted by veh# opsdate"

2.4.1.9 Deleting a Report

Reports may be deleted using the following command:

R> "Delete rows from reporter where rname = reportname"

2.4.2 Other Uses for Reports

In addition to the use of reports for performance measurement and vehicle schedules, a variety of other functions can also be generated as described below.

2.4.2.1 Ridership Analysis

As shown in the report called "trips" (Figure 2-29), ridership can be analyzed in any way chosen by the user. In this example, we have requested to see trips taken by clients classified as wheelchair elderly handicapped (weh). The command to generate this report is as follows:

R> "Print trips sorted by clientid tripdate where paxclass eq weh"

If we had attached no "where" condition, all trips taken during the specified time period would have been included.

This report illustrates the use of an attribute which can function as an internal counter. The attribute "trip#" assigns a numeric value to each record (1 = one-way trip; 2 = round trip). By simply defining a summation variable in reports ("sumtrips = sum of trip#"), R:base will calculate the total number of all trips, or trips disaggregated however we chose to do so.

2.4.2.2 Invoicing

Figure 2-30 demonstrates the use of reports to prepare an invoice for a human service program. This report, called "medbill", provides the information requested by the agency in order to reimburse the transit operation. It is sorted by medicaid# as this is most relevant to the recipient agency.* Total

*In order to produce this report, med# was added to the vsched file. We should have included it in the base files from the start.

trips are again calculated using the "trip#" attribute. Cost is determined by assigning a value to the cost of a trip and defining it as part of a variable in reports ("cost = sumtrips x 8.75"). The report can be generated by the following command:

```
R> "Print medbill sorted by med# tripdate where paycode eq med"
```

All the data shown on the report is generated by that command. We have entered the month "May" directly during the report design process. This can be easily edited at billing time each month.

Figures 2-38 and 2-39 demonstrate other examples of invoice reports. Figure 2-38 is an invoice sent to a city for trips taken by residents of the city and not covered by a human service program. Note the use of the footing to explain the various codes used on the invoice to its recipients. Note also that the name of the city, "Dennis", is entered by the report function as follows:

```
R> "Print citybill sorted by clientid tripdate where city eq +
R>  Dennis and paycode eq bill"
```

By changing the name of the city in the "where" clause, you can generate a distinct bill, properly labelled, for each political jurisdiction in your service area.

Figure 2-39 demonstrates an example of a client bill. Note again the explanatory material in the footing. A separate bill can be generated for each client by using "clientid" in the where condition as follows:

```
R> "Print bill sorted by tripdate where clientid eq 0001 and +
R>  paycode eq bill"
```

Note also in all three examples the use of the attribute "paycode" in the where clause to ensure that only trips billable to a particular institution or person are selected.

2.4.2.3 Disaggregating Performance Statistics

In the example shown in Figure 2-31 and used as the basis for Section 2.4.1, we chose to generate aggregate performance measurements for all vehicles. While this data is important, you will also probably want to view this information disaggregated by vehicle. This would enable you to isolate such events as poor load factors, high repair costs, poor fuel mileage, etc.

A weakness of R:base is the lack of a "subtotal" function which would enable you to accomplish this disaggregation on a single report. The only way to accomplish this function is to use the "where" clause to create a series of reports by vehicle. A sample is shown in Figure 2-40, created by the following command:

```
R> "Print vehstats sorted by opsdate where veh# eq A1"
```

CITY ASSESSMENT
City: Dennis
Month: May

ID	Last Name	First Name	Address	Tripdate	Destination	Trip#	Class	Purpose	Pay
0001	Baker	John	65 Bass	05/07/79	Pondville Hosp	2	e	hc	bill
0001	Baker	John	65 Bass	05/08/79	Pondville Hosp	2	e	hc	bill
0001	Baker	John	65 Bass	05/09/79	Pondville Hosp	2	e	hc	bill
0001	Baker	John	65 Bass	05/10/79	Pondville Hosp	2	e	hc	bill
0001	Baker	John	65 Bass	05/11/79	Pondville Hosp	2	e	hc	bill
0028	Morris	Ethel	2 Sweetwater	05/07/79	Dr. Fitch	2	e	hc	bill
0056	Boisvert	Mildred	3 Alford	05/11/79	Dr Brinkerhoff	2	e	hc	bill

TOTAL TRIPS: 14

TOTAL COST: 122.50

Purpose Code: hc = health care; nu = nutrition site; mow = meals on wheels; se = special education ; ft = field trips; gt = group trips; adc = adult day care; dl = dialysis.

Passenger Code: e = elderly; h = handicapped; eh = elderly handicapped; wh = wheelchair handicapped; weh = wheelchair elderly handicapped.

FIGURE 2-38. CITY INVOICE ("CITYBILL")

CLIENT BILL
Month: May

Name: Baker John ID: 0001
Address: 65 Bass City: Dennis

Trip Date	Destination	Purpose	Trip#	Paycode
05/07/79	Pondville Hosp	hc	2	bill
05/08/79	Pondville Hosp	hc	2	bill
05/09/79	Pondville Hosp	hc	2	bill
05/10/79	Pondville Hosp	hc	2	bill
05/11/79	Pondville Hosp	hc	2	bill

TOTAL TRIPS: 10

TOTAL COST: 10.00

Purpose Code: hc=health care; nu=site nutrition; mow=meals on wheels; se=special education; ft=field trips; gt=group trips; adc=adult day care; dl=dialysis

Trip# equals the number of trips taken on a given day. If you rode the bus both ways, trip# will equal "2". If you were accompanied by an escort, the escort's trips will also be counted.

FIGURE 2-39. CLIENT BILL ("BILL)"

VEHICLE STATISTICS

Veh#	Trip Date	Hours	Miles	Fuel	Cost	MPG	Oil	Cost	Repair	Cost
A1	05/07/79	10.0	249.0	10.9	8.500	22.84	2	2.16	radiator	45.00
A1	05/08/79	10.0	125.0	15.8	11.56	7.911				
A1	05/09/79	10.0	136.0	9.00	8.000	15.11				
A1	05/10/79	10.0	141.0	9.00	8.000	15.67				
A1	05/11/79	10.0	139.0	9.00	7.500	15.44				
Total:		50.0	790.000		43.5600	76.98				45.000
Average MPG:						15.4				
Miles/Trip:			4.115							
Trips/Mile:			0.243							
Trips/Hour:		3.840								

FIGURE 2-40. DISAGGREGATED VEHICLE STATISTICS

Before printing reports like this, however, you may need to modify some of the variables. As you recall, we generated the performance measurements in Figure 2-31 by using the total number of trips "480", as an absolute value in defining the report variables. This value must be changed to reflect trips associated with each vehicle.

First, use the "trips" report to generate this data as follows:

R> "Print trips where veh# eq A1"

Next, enter the D(efine) mode of reports and enter "R". The screen will list all variables and prompt you as follows: **variable name:** Enter the previously defined name of the variables which you want to change. You will receive the following prompt: **S(et), D(lete), Q(uit).** Type "S" and enter the new variable expression. In this example, we would substitute the value "192" (the number of trips accounted for by vehicle # A1) for the value "480" (the total number of trips).

2.4.2.4 Using Select to Generate Output

As discussed earlier (see Section 2.3.5), you can output data very simply by means of the select command. The disadvantage of select is that you have no way to design a clear and aesthetically pleasing report. However, in the case of a very small data base which changes rarely, the select function may be perfectly adequate. An example is demonstrated in Figure 2-41. This report is simply a listing of all vehicles owned by the agency. Given the amount and stability of the data, it hardly seems worth the effort to define a report. You will have to live with no titles or footings, and headings which are your attribute names. This "report" would be generated as follows:

R> "Select all from vmaster sorted by veh#"

veh#	vehid	lic#	yrmfg	make	model	engsz	title	lien	purdate	radio	access	#wchair
A1	-0	Bus207	1976	Dodge	B300	318CID	Elder Serv	EDTC	04/20/76	y	y	3
A2	-0	Bus208	1976	Dodge	B300	318CID	Elder Serv	EDTC	04/20/76	y	y	3
A3	-0	Bus209	1976	Dodge	B300	318CID	Elder Serv	EDTC	04/20/76	y	y	3
B4	-0	B37209	-0	Dodge	B300	360CID	Merchants	none	09/01/78	y	n	0
C2	-0	B70563	1974	Volks	wagon	UNK	Elder Serv	none	10/01/74	y	n	0

FIGURE 2-41. VEHICLE MASTER REPORT

2.5 OTHER APPLICATIONS SOFTWARE

This section describes the uses which can be made of three other types of applications software in paratransit management. These are spreadsheet, graphics, and word processing programs. As mentioned at the outset, the intent of this section is primarily to illustrate applications, and not to provide the type of detailed instructions provided in Sections 2.1 - 2.4. These programs are, for the most part, easy to use. The most detail is provided on the steps involved in using spreadsheets. This instruction is not specific to any one spreadsheet product since they all function in basically the same way. The major task in learning to use them is becoming familiar with their commands.

2.5.1 Spreadsheets

A spreadsheet is simply an electronic method of data manipulation. The spreadsheet program which we have used for this application is called "Perfect Calc". It is identical in function to such programs as "Visi Calc" and "Multiplan". Some spreadsheet programs, such as "Lotus 1-2-3", combine graphics and word processing capabilities in one integrated package.

A spreadsheet functions as a two-dimensional data array as shown in Figure 2-42 (see next page). Each array consists of vertical columns and horizontal rows. Perfect Calc has a single spreadsheet capacity of 50 columns and 250 rows of data.

The great advantage of a spreadsheet is its ability to instantly recalculate data. By constructing a series of simple arithmetic formulas, the spreadsheet will recalculate all designated values the moment you change one piece of data.

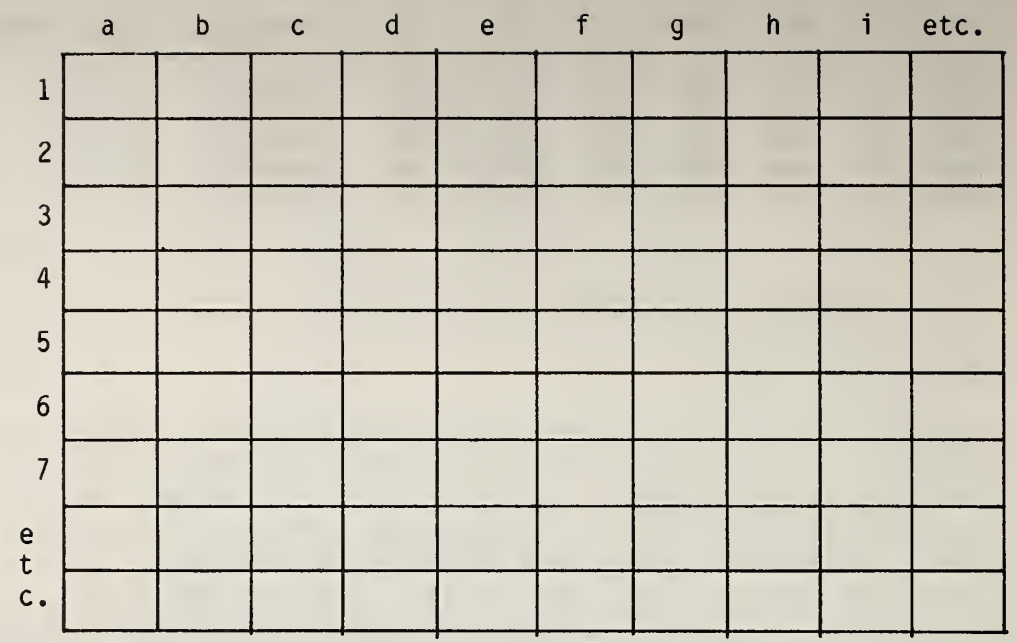


FIGURE 2-42. SPREADSHEET DESIGN

Below, we describe three spreadsheet applications: financial planning, budget management, and performance measurement. The most detailed explanation of the steps involved in using a spreadsheet is provided in Section 2.5.1.2, using spreadsheets for budget management.

2.5.1.1 Using Spreadsheets for Financial Planning

Figure 2-43 contains a spreadsheet which we have established to plan CAR's budget for the upcoming fiscal year. All the data is contained on a single spreadsheet and is completely interactive. This spreadsheet will enable the manager to conduct an endless series of "what if" scenarios in developing a budget. For example, suppose the manager does a first run-through and discovers a negative funding imbalance of $10,000. The manager might in effect say to him or herself "what if I could set an extra $10,000 from Title III, or what if I cut $10,000 from driver salaries"? By simply inputting these values, all totals will be recalculated on the spreadsheet.

There are four components to this spreadsheet model:

- I. Direct Expenses - Direct expenses are allocated among programs by function. In allocating these costs, the manager uses his or her operational knowledge to assign costs. For example, trips to nutrition sites are often much longer than trips to health care facilities. Therefore, a greater share of drivers costs are allocated

CAR PROGRAM BUDGETS

I DIRECT EXPENSES

Category	Nutri	Health	Therapy	ADC	Dialysis	Sp Ed	Med/Bos	Med/Pd	TOTAL
I. Direct Costs									
A. Personnel									
Dispatcher	2465	5164	377	913	216	806	1135	484	11560
Drivers	37039	19372	4820	12609	2313	10602	4280	6420	97455
Sub-Drivers	3602	4312	1338	2743	707	2027	302	529	15560
Fringe	6123	3803	806	2096	392	1768	1330	1261	17579
TOTAL A	49229	32651	7341	18361	3628	15203	7047	8694	142154
B. Other Direct									
Equipment	1440	800	200	520	120	480	160	280	4000
Gas/Oil	9000	5000	1250	3250	750	3000	1000	1750	25000
Maintenance	3600	2000	500	1300	300	1200	400	700	10000
Insurance	10187	5659	1415	3679	849	3396	1132	1981	28298
Licenses	117	65	16	42	10	39	13	23	325
Communications	2460	2888	450	617	430	693	552	470	8560
Space	1800	1000	250	650	150	600	200	350	5000
Utilities	900	500	125	325	75	300	100	175	2500
									0
TOTAL B	29504	17912	4206	10383	2684	9708	3557	5729	83683
TOTAL DIRECT (A&B)	78733	50563	11547	28744	6312	24911	10604	14423	225837

FIGURE 2-43. FINANCIAL PLANNING

IIa VEHICLE HOUR CHART

Description	Nutri	Health	Therapy	ADC	Dialysis	Sp Ed	Med/Bos	Med/Pd	TOTAL
A1 Dodge Maxi (76) (lift)		5.0		5.0					10.0
A1 Dodge Maxi (76) (ramp)		5.0				3.0			8.0
A3 Dodge Maxi (76) (lift)		3.0	5.0	2.0					10.0
A4 Dodge Maxi (76) (ramp)		5.6		2.0	2.4				10.0
B1 Dodge Maxi (79) (lift)	5.0			3.0					8.0
B2 Volks Bus (74) (7)	5.0								5.0
B3 Dodge Maxi (77) (12)	4.5					5.5			10.0
B4 Dodge Maxi (77)	5.0					2.0			7.0
B5 Dodge Maxi (77) (12)	6.0								6.0
C1 Dodge Maxi (76) (12)	7.0								7.0
C2 Volks Bus (74) (7)								6.0	6.0
C3 Dodge Maxi (77) (lift)							4.0		4.0
SPARE									
SPARE									
Total Vehicle Hours	32.5	18.6	5.0	12.0	2.4	10.5	4.0	6.0	91.0
% of Total Vehicle Hrs	35.71	20.44	5.49	13.19	2.64	11.54	4.40	6.59	100.00

IIb INDIRECT EXPENSES

Category	Nutri	Health	Therapy	ADC	Dialysis	Sp Ed	Med/Bos	Med/Pd	TOTAL
II. Indirect									
A. Personnel									
General Manager	6889	3943	1060	2544	509	2226	848	1272	19289
Financial Manager	4718	2700	726	1742	348	1524	581	871	13210
Program Assistant	3577	2047	550	1321	264	1156	440	660	10016
Clerk/Typist	3297	1887	507	1217	243	1065	406	609	9231
Training	263	154	41	99	20	87	33	50	754
TOTAL A	18750	10731	2885	6923	1385	6058	2308	3462	52500
B. Other									
Travel	357	204	55	132	26	115	44	66	1000
Printing/Supplies	714	409	110	264	53	231	88	132	2000
Advertising	179	102	27	66	13	58	22	33	500
Legal/Audit	1071	613	165	396	79	346	132	198	3000
Postage	286	164	44	105	21	92	35	53	800
TOTAL B	2507	1492	401	963	193	842	321	481	7300
TOTAL INDIRECT (A&B)	21357	12223	3286	7886	1577	6900	2629	3943	59800

FIGURE 2-43. FINANCIAL PLANNING (Cont'd)

III INCOME

Funding Source	Nutri	Health	Therapy	ADC	Dialysis	Sp Ed	Med/Bos	Med/Pd	TOTAL
Title III		20000							20000
Title VII	72900								72900
Title XIX		10000	10000	18000	6000		2000		46000
Title XX		18000	1500	1500	1000				22000
Nursing Homes				12000					12000
Schools						27000			27000
CETA	6575						5490	7630	19695
Donations	5000		2000	2000	1000		6000	5000	21000
Rehabilitation						2000			2000
CCRTA	18685	20357							39042
Other								4000	4000
									0
TOTAL INCOME	103160	68357	13500	33500	8000	29000	13490	16630	285637
% of Total Income	36.12	23.93	4.73	11.73	2.80	10.15	4.72	5.82	
TOTAL EXPENSES(I&IIb)	100090	62786	14833	36630	7889	31811	13233	18366	285637
% of Total Expenses	35.04	21.98	5.19	12.82	2.76	11.14	4.63	6.43	
BALANCE	-3070	-5571	1333	3130	-111	2811	-257	1736	0

FIGURE 2-43. FINANCIAL PLANNING (Conc'd)

to nutrition trips. On the other hand, since most nutritional trips are prescheduled, they are less of a burden on dispatchers than are health care trips. Both the horizontal and vertical totals are automatically calculated by the spreadsheet as a result of simple summation formulas created by the user.

- II (A & B) Vehicle Hour Chart and Indirect Expenses - These two parts of the spreadsheet are interactive with each other. In IIa, the manager allocates each vehicle's service hours to specific programs. Through the use of a formula, the spreadsheet will calculate the percentage of total vehicle hours accounted for by each program. These percentages are then used to calculate indirect expenses on part IIb. The manager first determines the total cost of each function. Since indirect expenses do not vary by program like direct expenses (the General Manager does not necessarily spend more time on one program than another), it does not make sense to individually assign these costs. Instead, each program is assigned the percentage of indirect costs represented by their percentage of total vehicle hours. Thus, nutrition accounts for 35.71% of the vehicle hours, it will account for the same percentage of each indirect expense item. These values will be automatically calculated by applying in a formula the appropriate percentage to the total cost of each item.

- III. Income - The final component of this spreadsheet performs the same type of calculation on the income side of the ledger. Then, income can be contrasted with expenses. The spreadsheet will combine the values previously determined for total indirect and direct expenses, and then compare this new value to the total income line. The bottom line is the balance. As one would hope in establishing a budget, the value in the lower right-hand corner is "0", meaning that total anticipated expenses equal total income. In reaching this figure, however, you can see along the bottom line that each program is slightly out of balance to some extent. This is inevitable, although the manager might well be concerned about the $5571 deficit in health care financing.

Once established, this spreadsheet can be used year after year by simply changing the values as needed. Through simple commands, new lines or columns can be added or deleted. It is also easy to alter the width of columns.

2.5.1.2 Using Spreadsheets for Budget Management

The spreadsheet developed in Section 2.5.1.1 was a relatively static document, developed once a year and then set aside until the following year. The spreadsheet shown in Figure 2-44 is an active document to be employed during the course of a fiscal year to track the agency's progress toward the previously established budgetary goals. A more detailed explanation of how to construct and use this spreadsheet is provided below.

The construction of the spreadsheet in Figure 2-44 is a relatively straightforward task. The first step, as in the development of a data base, is to conceptualize the substance and design of the spreadsheet. In this case, our

CAR INCOME EXPENSE REPORT APRIL

Category	Health Care	Therapy	Adult Day Care	Dialysis	Nutrition/MOW	Special Ed	Nursing Home	Other	Monthly Total	Prior Total	New Total	Budgeted Amount	Balance
Income													
Elder Services(III & VII)	2640.65				7522.55				10163.20	18800.00	28963.20	114900.00	85936.80
Title XIX	1000.00	1000.00	2000.00	500.00					4500.00	20234.42	24734.42	46000.00	21265.58
Cape Cod Collab (Schools)						3874.40			3874.40	16027.28	19901.68	27000.00	7098.32
Nursing Homes							57.00		57.00	2177.51	2234.51	12000.00	9765.49
Foundations(Donate)							950.00		950.00	2400.00	3350.00	21000.00	17650.00
Contributions (Other)								50.00	50.00	628.00	678.00	4000.00	3322.00
CCRTA									0.00	71148.27	71148.27	39042.00	-32106.27
CETA									0.00	0.00	0.00	19695.00	19695.00
Rehabilitation									0.00	0.00	0.00	2000.00	2000.00
Total	3640.65	1000.00	2000.00	500.00	7522.55	3874.40	1007.00	50.00	19594.60	131415.48	151010.08	285637.00	134626.92
Expenses													
Salaries	2332.64	878.93	1597.07	691.72	4141.10	1591.40	64.29		11297.15	74801.00	86098.15	177075.00	90976.85
Fringe Benefits	411.18	186.17	196.18	71.18	993.65	20.82			1799.18	7167.00	8966.18	17579.00	8612.82
Postage	26.36	0.48	1.00	0.27	0.00	0.00			28.19	400.00	428.19	800.00	371.81
Printing & Office	8.92	0.64	1.51	0.38	22.65	23.43			57.53	1378.00	1435.53	6000.00	4564.47
Insurance	296.80	0.00	0.00	0.00	0.00	333.94			630.74	15694.00	16324.74	28298.00	11973.26
Vehicle Gas & Oil	0.00	138.52	134.32	142.07	0.00	323.32			738.23	14053.00	14791.23	25000.00	10208.77
Vehicle Maintenance	0.00	377.01	425.68	330.51	707.01	337.57			2177.78	13682.00	15859.78	10000.00	-5859.78
Travel	4.50	0.00	0.00	0.00	0.00	0.00			4.50	426.00	430.50	1000.00	569.50
Utilities	12.60	4.23	10.04	2.49	30.25	9.64			69.25	1588.00	1657.25	2500.00	842.75
Rent(space)	72.00	24.40	58.00	14.40	174.00	55.60			400.00	2398.00	2798.00	5000.00	2202.00
Telephone	270.46	47.93	202.40	1.95	544.42	358.67			1425.83	5384.00	6809.83	8560.00	1750.17
Other(Audit,Adv,Lisc)	0.00	0.00	0.00	0.00	0.00	0.00			0.00	14.70	14.70	3825.00	3810.30
TOTAL	3436.26	1578.31	2626.28	1254.97	6613.88	3854.39	64.29	0.00	18628.38	136985.70	155614.08	285637.00	130022.92
BALANCE	204.39	-578.31	-626.28	-754.97	908.67	820.01	942.71	50.00	966.22	-5570.22	-4604.00	0.00	-4604.00

FIGURE 2-44. BUDGET MANAGEMENT

objective is to track income and expenses by transportation category. We want to accomplish this tracking on a monthly and year-to-date (YTD) basis on the same spreadsheet.

Most spreadsheet programs offer a variety of design options which can be implemented by simple commands. For example, in Perfect Calc, the user may 1) change the width of columns; 2) justify (left or right) or center entries; and 3) determine the display format (i.e., number of decimal places, etc.). In this example, the "category" column has been widened; the column headings have been centered; and a standard two decimal display format selected. Ideally, the user should make this type of design selection at the beginning of the process. Changes can, however, be made at any time either globally (throughout the spreadsheet) or column/line specific.

The user enters data into the spreadsheet simply by typing the appropriate letters/numerals. If the first digit is a letter, the program will automatically assume that a "label" is being entered. If the first digit is a numeral, the program will assume "number". The user should follow specific spreadsheet conventions for combining letters and numerals in a single data entry.

In Perfect Calc, active data entry appears in the lower left-hand corner of the screen. This entry can be inserted into the spreadsheet by hitting "return" or any of the cursor control keys. The entry will immediately reappear on the spreadsheet at the position where the cursor had been located.

There are several common spreadsheet features of which users should be aware. As spreadsheets grow in size, the user will need to "move around" the spreadsheet quickly and efficiently. This can be accomplished either by moving the entire spreadsheet so that a different portion is visible on the screen (or "window"), or by moving the cursor to different locations on the spreadsheet. Spreadsheets can be moved up and down, or side to side. The cursor can usually be moved to the beginning or end (or top or bottom) of lines and columns, or to any specific location on the spreadsheet. These movements can be accomplished by means of single commands. Perfect Calc has the added feature of being able to divide the screen in half either vertically or horizontally. In Figure 2-44 this would permit the user to view the "category" and "new total" columns simultaneously (for example).

Efficient editing is a major selling point of spreadsheets. Columns and lines can be added or deleted, causing the rest of the spreadsheet to adjust in response. Data entry items can be edited so that a single digit can be corrected without retyping the whole item. Data items can also be easily deleted.

Spreadsheets perform their basic computational function by means of simple arithmetic formulas inserted by users. To create a formula, it is necessary to enter the formula "mode". In Perfect Calc, this is accomplished by entering "=". Formulas are created by means of column and line coordinates. For example, in Figure 2-44, the amounts in the "Monthly Total" column would be obtained by summing the values in the previous columns. In Perfect Calc, columns are referenced by letters and lines by numerals. Thus, the first Monthly Total amount ($10163.20) appears in column "j" on line "7" (including

blank lines, underlining, and titles). Thus, by positioning the cursor at this location and entering the formula mode, the user will receive the following prompt: j7=. The user would then enter the following formula: "sum (b7:i7)". This formula, in Perfect Calc, instructs the program to add all the values between "b7" and "i7".

Of course, you will want to replicate this formula across each line of data. Again, this can be easily accomplished by means of a series of commands. The formula does not need to be reentered line-by-line by the user.

This spreadsheet is intended to assist the manager in tracking funds on an accrual basis. An important feature of managing a government funded program is the wide disparity in time between when income and expenses are "accrued" and when they are translated into "cash-in" and "cash-out". This makes it difficult for the manager to track progress toward predefined budgetary goals.

The spreadsheet in this example enables the manager to track expenditures both by transportation program (the vertical columns) and by budget line item (the horizontal rows). One can observe several significant trends on this spreadsheet. Most significantly, expenses are outstripping income by $4604 as of April. (The CAR fiscal year ran from October 1 to September 30, so this represents one month more than half a year.) Contributions from Elder Services ($28,963) are far behind the projected pace ($114,900). This gap has been filled in by the CCRTA which has already provided almost double ($71,148) its anticipated contribution ($39,042). No funding at all has been received from CETA and Rehabilitation.

On the expense side, salaries are running slightly behind the budgeted pace ($86,098 to $177,075 for the whole year). On the other hand, the energy crisis of 1979 has impacted fuel costs (already $14,791 out of $25,000 budgeted) and the age of CAR's fleet has finally caught up with it as maintenance costs ($15,859) have already exceeded the budgeted amount ($10,000). A manager, observing these latter two trends, might have taken the step of reducing labor costs to stay within the total budget. In this way, managers are able to know at all times where they stand in relation to overall budgetary goals and constraints.

The spreadsheet in Figure 2-45 performs a similar function by looking at a different set of categories. This spreadsheet tracks actual cash flow, as opposed to accrued expenses. Like the spreadsheet in Figure 2-44, it can be created once at the beginning of the fiscal year with new data added each month. Assets and liabilities will be totalled each month and compared in the "cash balance" line. The cash balance for all of the months will be summed and reflected in the bottom line "net balance for year".

2.5.1.3 Using Spreadsheets for Performance Measurement

The spreadsheet shown in Figure 2-46 uses some of the same statistics developed in the R:base report called "vehstats".

CAR CASH BALANCE

Category	October	November	December	January	February	March	April	May	June	July
Current Assets										
Checking Account	3330.84	2763.68	381.89	8467.94	6586.74	5872.09				
Savings Account	5187.08	4253.84	4261.84	306.14	397.18	508.53				
Petty Cash	30.00	30.00	30.00	30.00	30.00	30.00				
Fixed Assets										
Office Equipment	1610.00	1610.00	1178.84	1178.84	1178.84	1178.84				
Vehicle Equipment	479.17	479.17	354.17	354.17	354.17	354.17				
Other Assets										
Security Deposits	1688.00	1688.00	1369.00	1369.00	1369.00	1369.00				
Accounts Receivable	14767.00	9588.00	31807.00	28261.00	28361.00	49850.00				
TOTAL ASSETS:	27092.09	20412.69	39382.74	39967.09	38276.93	59162.63				
Current Liabilities/Funds										
Payroll Taxes Wh	1629.10	1763.10	1598.34	1607.57	1660.92	2429.71				
Adv from Elder Serv	6000.00	6000.00	6000.00	6000.00	6000.00	6000.00				
Uncollectable Medicaid	2573.08	2573.08	1671.49	1671.49	1671.49	1671.49				
Capital Equip Fund	10000.00	10000.00	10000.00	10000.00	10000.00	10000.00		10000.00	10000.00	10000.00
Accounts Payable			6067.64		2995.00	14240.00				
Loan Payable	5960.80	5960.80	12206.92	12206.92	12206.92	17206.92				
Advance from CCRTA						13900.00				
TOTAL LIABILITIES/FUNDS	26162.98	26296.98	37544.39	31485.98	34534.33	65448.12				
CASH BALANCE	929.11	-5884.29	1838.35	8481.11	3742.60	-6285.49				
NET BALANCE FOR YEAR	2821.39									

FIGURE 2-45. BUDGET MANAGEMENT - II

VEHICLE STATISTICS MAY

Vehicle #	Hours	Miles	Fuel	Oil	F&O Cost	MPG	Rep Cost
A1	50.00	790.00	53.70	2.00	45.72	14.71	45.00
A2	42.50	1273.00	89.60	2.00	73.58	14.21	
A3	39.50	762.00	93.50		75.85	8.15	25.75
B4	32.50	484.00	43.20		35.50	11.20	100.00
C2	39.75	1016.00	59.70	2.00	48.60	17.02	
Total	204.25	4325.00	339.70	6.00	279.25	12.73	170.75

FIGURE 2-46. PERFORMANCE MEASUREMENT

As should be apparent by now, our spreadsheet program is far more nimble and versatile at performing arithmetic calculations than is R:base. Other data base managers such as Knowledgeman are far stronger in this area than is R:base, but they tend to be much harder to use. The user must determine which characteristic is more important for their own uses, and how best to perform each function. No data base manager is as good at calculations as spreadsheet programs unless, as in the case of Knowledgeman, they integrate a spreadsheet right into the program.

The extent of integration between a data base manager and a spreadsheet program depends on the characteristics of each program. If integration requires little manipulation, it may be worthwhile to attempt to transfer data directly between the two programs. That was not the case with R:base and Perfect Calc, however, so we took this data off of R:base reports and entered it manually into Perfect Calc. On the other hand, our word processing program, Perfect Writer, works quite well with R:base and we provide in Section 2.5.3 an example of moving data directly between these two programs.

2.5.2 Graphic Applications

The next two figures demonstrate the tremendous potential of graphics software. In Figure 2-47, we have used our graphics program to produce a pie chart of some data generated by the R:base report "trips". As you recall (see Section 2.4.2.1), this report could generate ridership statistics disaggregated in any fashion we chose. In this case, we have analyzed the data by passenger classification. By entering this data into Fast Graphs, we were able to obtain a graphic portrayal of CAR's ridership characteristics for the case study week of May 7-11, 1979.

The three dimensional bar graph in Figure 2-48 is based on the Perfect Calc financial planning spreadsheet displayed in Figure 2-43. This file assigned percentages of total expenses, total income, and vehicle hours to each program. As we can see in this graph, there are no gross anomalies present in the allocation of costs, expenses and vehicle hours.

Again, the degree of integration between graphics and other applications software varies greatly. In most cases, Fast Graphs can read a Perfect Calc file. However, despite being sold as an integrated package, we discovered

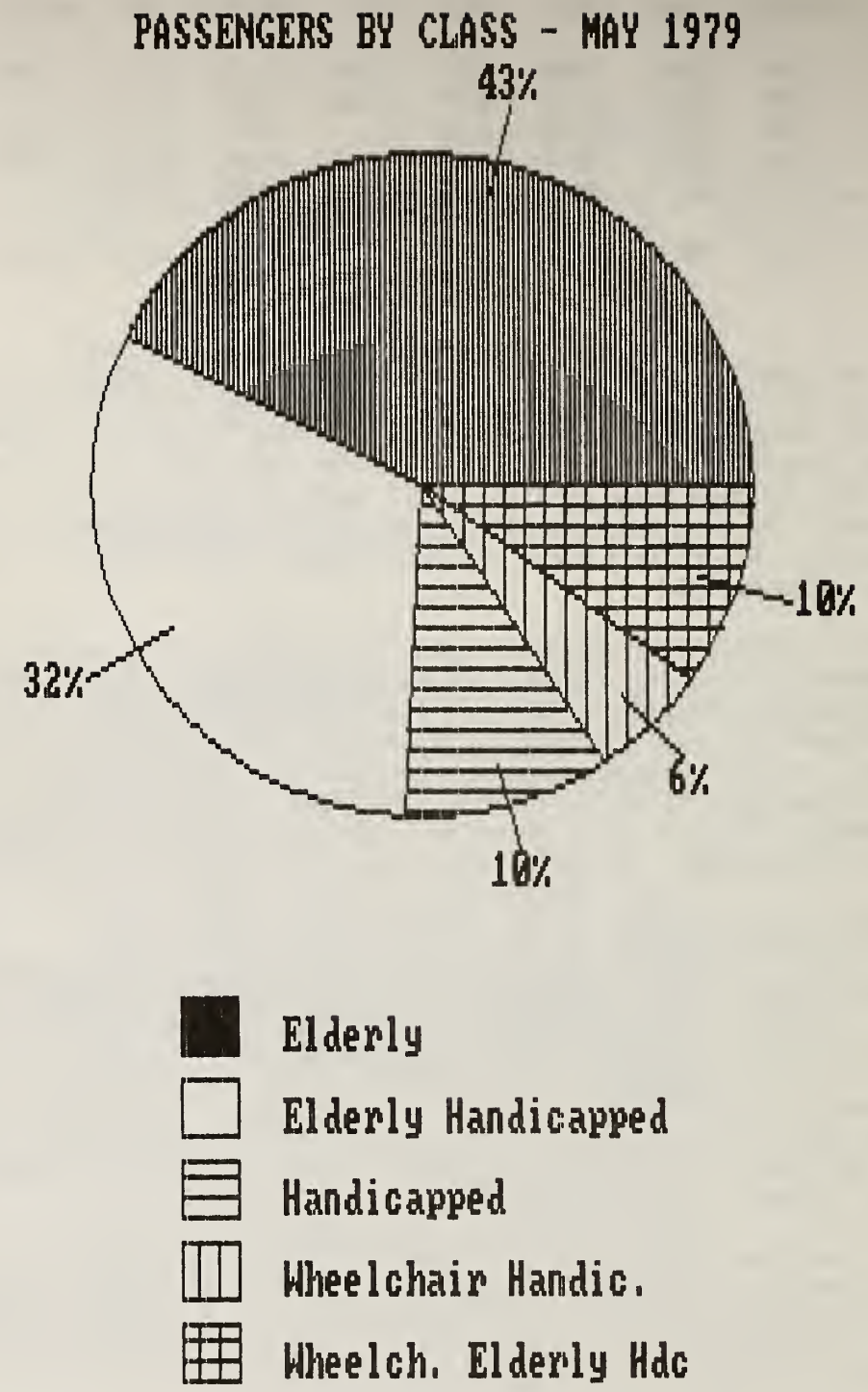


FIGURE 2-47. CAR'S RIDERSHIP CHARACTERISTICS

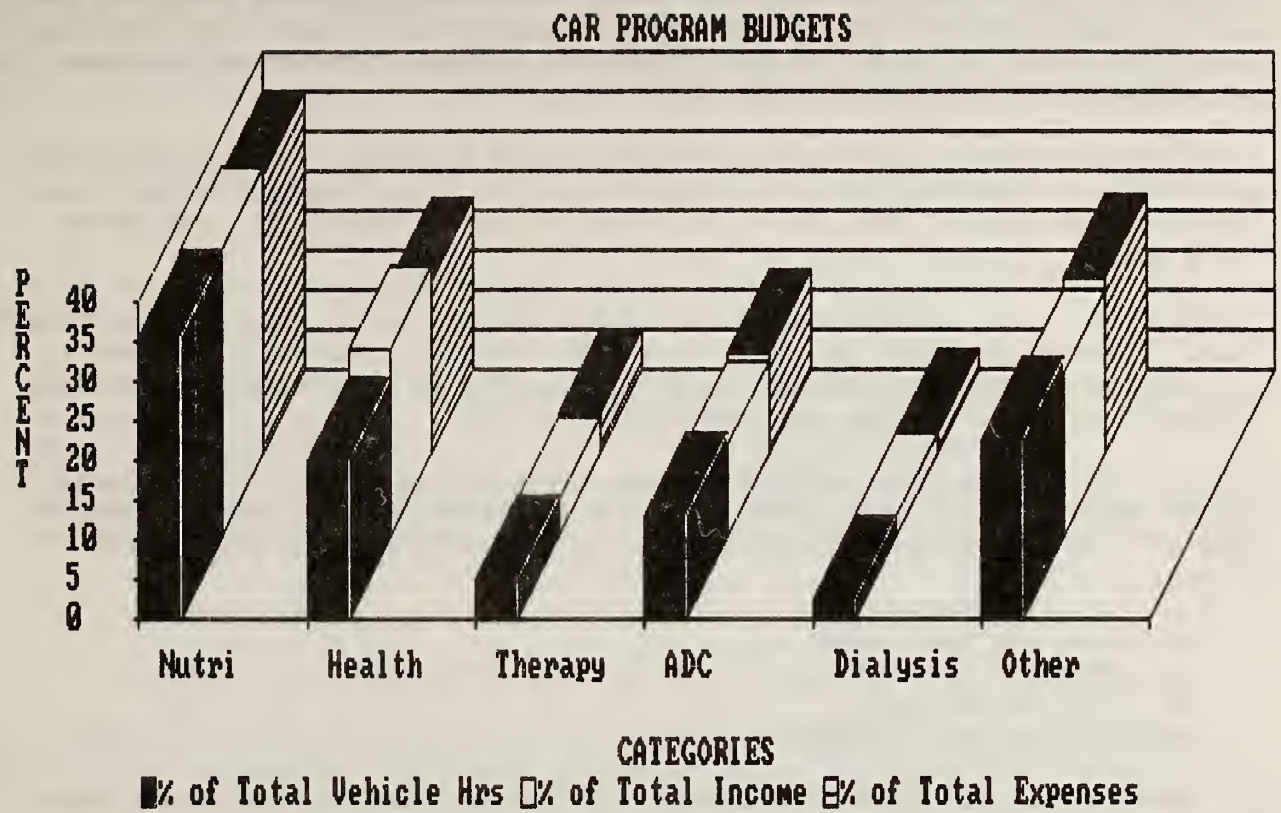


FIGURE 2-48. COMPARING CAR'S INCOME/EXPENSE/VEHICLE HOURS BY PROGRAM

that Fast Graphs was not compatible with our printer (which is compatible with Perfect Calc), and could not read Perfect Calc values which had been generated by formula as opposed to direct data entry. These problems have been corrected in an updated version.

Lotus 1-2-3 provides a completely integrated spreadsheet and graphics capability, but a far weaker word processing program than Perfect Writer (see the following section).

2.5.3 Integrating R:base with a Word Processing Program

Since R:base has the capability of writing ASCII data files (using the "output" command), most word processing programs can automatically (i.e., without any modification) read those files, and manipulate them using its own set of command. Thus, data stored in the R:base master client file, such as names, addresses and other client information, can be transferred to a word processing program without retyping the data.

This characteristic enables us to integrate both programs in order to produce personalized documents that are basically created using "Perfect Writer", but that also use sets of data (or a few pieces of data) created by, and stored in, R:base.

Unfortunately, the integration is not always 100% perfect. Some manipulation (i.e. "reformatting") of the file created by R:base is necessary to adapt it to the needs of the word processing program. This is the case in the example presented below, using the program "Perfect Writer".

Suppose that our purpose is to write some personalized letters to all those CAR patrons whose Title XX units are about to expire, and we want to inform them of this situation. We need the following pieces of information from R:base:

- first and last names
- address
- city and zip code
- Title XX units remaining

All these items are stored in R:base in the relation called "msclient" under the attribute names "frstname", "lastname", "address", "city", "zip" and "xxunits" respectively. Use the "output" and "select" commands (inside R:base) as shown in Figure 2-49 to write an ASCII file called "b:text.mss" that contains all those fields with the information for which you are looking. This file is immediately accessible by Perfect Writer, typing the command (in MS-DOS): "A> "pw b:text.mss".

```
R> output b:text.mss with terminal
R> select frstname lastname address city zip xxunits +
R> from msclient where xxunits le 02
```

FIGURE 2-49. FILE OUTPUT COMMAND

File output is shown in Figure 2-50:

frstname	lastname	address	city	zip	xxunits
Claire	O'Rourke	Windmill Villag	Dennis	02638	02
Ethel	Downey	109 Upper Ct	Dennis	02638	02

FIGURE 2-50. SAMPLE OUTPUT FILE

Once inside Perfect Writer (and inside the b:text.mss file), we manipulate the file, using Perfect Writer commands, until the appearance of it is as shown in Figure 2-51. In doing so, we have created two separate files called b:test1.mss and b:test2.mss. Each file contains the data for one client. The manipulation consists of using the perfect writer editing functions to group the words and sentences, and adding the following Perfect Writer commands:

- String
- Flushleft
- Value

A brief description of these Perfect Writer commands is provided at the end of this Section.

```
@STRING(address="Claire O'Rourke
Windmill Village
Dennis, MA 02638")
@STRING(xx="02")
@STRING(name="Ms. O'Rourke")
@FLUSHLEFT{@VALUE(address)}
```

b:test1.mss

```
@STRING(address="Ethel Downey
109 Upper Ct.
Dennis, MA 02638")
@STRING(xx="02")
@STRING(name="Ms. Downey")
@FLUSHLEFT{@VALUE(address)}
```

b:test2.mss

FIGURE 2-51. REARRANGING THE OUTPUT FILE WITH PERFECT WRITER

Once these two files have been created, we also use Perfect Writer to create the body of our letter. The letter is contained in the file "b:mail.mss". This letter is shown in its preformatted Perfect Writer format in Figure 2-52. With this letter edited, our task is completed. The result, a personalized letter, is shown in Figure 2-53. The name and address of the client, as well as the number of Title XX units remaining were taken directly out of the R:base relation "msclient".

Following is a brief description of the Perfect Writer commands which appear in Figure 2-52.

- Include (filename.mss) instructs Perfect Writer to insert the file named between the parentheses into the text of the file being printed. Don't forget the disk drive extension "b:". Following the insertion of the file defined in this manner, Perfect Writer continues printing the letter file "b:mail.mss". Our one-time R:base file is now inserted into this Perfect Writer file.

```
@PAGEFOOTING()
@INCLUDE(b:test2.mss)

@FLUSHRIGHT(April 1, 1984)


@FLUSHLEFT(Dear @VALUE(name):)

This letter is to inform you that you have only @VALUE(xx) Title
XX units left for your use in utilizing our services.

Since we can not provide you with more transportation services
once the units are totally consumed, we urge you to contact your
Human Services Representative to authorize more Title XX units
for your use.

We thank you for your understanding, and hope we will hear from
you soon so that we may continue to meet your transportation needs.

@FLUSHLEFT(Sincerely yours,)

Perfect Writer Version 1.00 (Wrap) mail: B:MAIL.MSS -0%-
(c) 1983, Perfect Software Inc. -- Type ESC ? for help
```

FIGURE 2-52. SAMPLE LETTER WITH PERFECT WRITER COMMANDS

April 1, 1984

Dear Ms. :

This letter is to inform you that you have only 02 Title XX units left for your use in utilizing our services.

Since we can not provide you with more transportation services once the units are totally consumed, we urge you to contact your Human Services Representative to authorize more Title XX units for your use.

We thank you for your understanding, and hope we will hear from you soon so that we may continue to meet your transportation needs.

Sincerely yours,

General Manager
CAR

FIGURE 2-53. FINAL SAMPLE LETTER

- String/Value (variable name - string text). The string command (as shown in Figure 2-51) assigns a definition (string text) to a variable name. The value command inserts that definition into the text. Thus, values have been assigned in Figure 2-51 to the variable names "xx" and "name". The value for "name" is inserted into the salutation of the letter. The value for "xx" (the number of remaining Title XX units for the addressee) is inserted directly into the appropriate location in the body of the text.

- Pagefooting suppresses the printing of page numbers.

- Flushleft aligns the text enclosed by parentheses to the left-hand margin.

3.0 USER OPTIONS

The purpose of this chapter is to provide the user with a variety of options for the use of data base management software in paratransit management. Section 3.1 below ("Variations on Chapter 2.0 Applications") describes some options which the user might want to consider in designing their data base management system. This section covers only those applications discussed in Chapter 2.0 ("Automating the CAR data base"). Of necessity, instructions for implementing these options are brief and completely non-R:base specific. The user who has absorbed the information in Chapter 2.0 should be able, with the assistance of a specific software manual, to apply these options using any similar piece of software.

Section 3.2 below ("Other Data Base Applications") is intended to provide the user with a brief overview of some other potential applications which were not discussed in Chapter 2.0

3.1 VARIATIONS ON CHAPTER 2.0 APPLICATIONS

In this section, we discuss options based on the general operating procedures of the user agency; the attributes (fields) available for inclusion in relations (files); and desired reporting requirements.

3.1.1 General Operating Procedures

Four issues present themselves regarding the general operating procedures of the user agency: 1) the type of transportation service being offered; 2) service level requirements; 3) the location of the scheduling/dispatching function; and 4) the degree of interaction with a live customer on the telephone. Each is discussed below.

3.1.1.1 Service Type

While Chapter 2.0 was written specifically for the standard demand-responsive model of many origins and many destinations, it could be easily adapted to any client-specific paratransit mode. For example, checkpoint and route deviation have become increasingly popular hybrid models between the standard forms of demand-responsive and fixed-route service. In checkpoint service, some passengers are picked-up at designated gathering spots (i.e., bus stops), while others are picked-up in a demand-responsive mode. Similarly, route deviation services offer to deviate off of a fixed-route to pick-up passengers in response to phone-in demand.

In both cases, the pre-scheduled files used in Chapter 2.0 could be replaced with route-specific schedule files. These files would provide a dispatcher with a written "picture" of the standard route to be operated by each vehicle. Information might include scheduled stops, roads traversed, major activity centers, etc. When a demand-responsive call is received, the operator, instead of checking the prescheduled file for vehicle availability as in Chapter 2.0, would check the vehicle schedule files and book the trip accordingly. A report could then be generated in the same way as in Chapter 2.0, showing each driver their particular routing for the day.

3.1.1.2 Service Level Requirements

A great deal of emphasis is placed in Chapter 2.0 on the advantages of prescheduling standing trips to take maximum advantage of automated scheduling systems. There are two reasons for this recommendation. First, by prescheduling the bulk of trips, a stable route structure can be developed which is familiar to customers, drivers, and dispatchers alike. This tends to assure the most reliable service for the most regular customers, a worthy goal of any business. Second, by requiring a minimum of 24-hour in-advance reservations for non-prescheduled trips, the user will have sufficient time to solidify routings for the next day and provide each driver with a hard copy schedule print-out.

Before undertaking the use of data base management software for scheduling purposes, the user must confront the issue of service level requirements. There are two predominant trends in the paratransit field today. One theory advocates ever increasing advanced scheduling to better match the service predictability and reliability of the fixed-route. This tendency accounts for the increasing popularity of the checkpoint and route deviation models discussed above. The other theory advocates the elimination of all prescheduling requirements to better emulate the spontaneous nature of riding a fixed-route. This latter approach requires a far greater investment in the scheduling process in order to maintain desirable vehicle load factors and schedule predictability.

The choice of one model over the other depends on which element of the fixed-route service the user values most highly - predictability/reliability, or the lack of advanced planning needed by a customer to use it. We think the choice may depend on the agency's client base. The prescheduling model best serves the regular rider. The occasional rider must fit into a schedule built around the needs of the regular users. This model is most appropriate for the system which is still primarily client/agency specific (closer to a Section 16(b)(2) system than a true Section 18 general public system). Most of this system's riders will be regular users and should receive priority service.

The real time model places the occasional user on an equal footing with the regular riders. This is clearly more appropriate as the system moves toward a more truly "open to the general public" status.

Both models require the user to make several choices prior to automating the scheduling function. First, you must select a cut-off time for real time trip reservations. Given the preference of our case study operator for the prescheduling model, we imposed the standard 24-hour advance reservation requirement in the system we established in Chapter 2.0. This is really the bare minimum amount of time needed to print-out driver schedules at the end of each day. In a system with more than a few vehicles, this could prove quite tense. If the emphasis is on prescheduling, we recommend considering a 48-hour cut-off. However, if the emphasis is on obtaining true "real time" operation, you may want to eliminate advance reservation requirements entirely.

Second, you need to consider how to group your prescheduled trips. In Chapter 2.0, we recommend two options; each day of the week or a MWF/TTh pattern. Of course, any such pattern is feasible given the demands of your system. You might also want to consider incorporating an "every other week" element if that fits your service pattern. The simpler the pattern (such as MWF/TTh), the easier it is to perform the prescheduling function (since you need to create fewer unique files). However, if your trip pattern does not conform to such a simple model, you won't gain anything by trying to over-simplify it.

The third factor to consider is how far in advance to preschedule. Operationally, the further in advance, the better (although we don't recommend more than a month because there are apt to be too many changes). In Chapter 2.0, we discussed prescheduling at one week and one month intervals. Since you will need to maintain a preschedule file for each active date, the further in advance you preschedule, the greater will be the demand on your disk storage space. It is hard to imagine prescheduling for a month without hard disk capability.

The real time scheduling model also presents the user with several options. Given the reduced amount of routing regularity inherent in this model, more pressure will be placed on the operator to "hunt-out" the best vehicle schedule among several likely options. Since you will want to accomplish this search interactively with the customer on the telephone, it must be accomplished rapidly. We strongly recommend that the schedule search be customized so that the operator can simply respond to a series of menu-driven prompts. See Appendix C for a description of customizing R:base.

The user must also consider the necessity of a high-quality communication system so that dispatchers can inform drivers of real time schedule changes.

3.1.1.3 Location of Scheduling/Dispatching Functions

The user needs to evaluate where to locate the scheduling and dispatching functions. Chapter 2.0 was written for the small system with a single user station. The user may require more than one station at a single site to handle demand. If the agency provides transportation over a large service area, it may want to decentralize scheduling and dispatching. Similarly, if service is provided by a consortium of providers, these functions may be institutionally decentralized.

This issue will lead the user to a consideration of the potential and desirability of a multi-user system. Multi-user applications with microcomputers (as opposed to minicomputers and mainframes), is still in its infancy. Nevertheless, if your demand is sufficient to warrant providing several clerks/dispatchers with simultaneous access to the data base, it is an issue which must be addressed.

Multi-user capability requires a hard disk. A number of data base software manufacturers have adapted their products for use in limited multi-user environments. These include R:base 6000 and DB Master Advanced Edition. (DB Master is a File Manager, not a relational data base.) So far, most microcomputer multi-user applications include "read only" capability (hence, the term "limited"). This means that a variety of users can access the system

and "read" data simultaneously, but they cannot "write" (i.e., update a file, schedule a trip, etc.) simultaneously. A system of this type has been installed at the Portland (Maine) Rural Transportation Program using DB Master Advanced Edition.

Given the size of most rural paratransit operations, this limited capability may be all that is necessary or desirable. Unless the system has an unusually high volume of telephone calls, there will usually not be a problem in having a clerk read the client's file to check for eligibility while waiting for another user to relinquish control of the "write" facility. In fact, this capability of "shutting out" other users who want to write (the program will not allow it) may actually be a positive feature in a small system where multi-user writing might prove more confusing than helpful.

This is an area witnessing rapid technological advance so the user should be willing to investigate the latest developments. It is important to remember, however, that what becomes technologically possible may not necessarily be operationally desirable. The user must determine what degree of interaction is required for effective operation among the system's sites. For example, leasing telephone lines for continuous communications can prove expensive. In a small system, sporadic (dial up) communications may prove adequate if the remote sites operate relatively independently with little need to coordinate data on a regular basis.

3.1.1.4 Customer Interaction

A final operational decision facing the user is which functions should be accomplished interactively with a customer on the telephone. The options include 1) checking eligibility, 2) booking a trip, and 3) updating the master client file.

As described in Chapter 2.0, we recommend performing all three functions interactively. The advantage of this approach is that it eliminates time-consuming "middle steps" such as recording data temporarily either manually or on the computer for later integration into the main files; and calling back customers to inform them of the outcome of their trip request (either the details of the booking or the unfortunate fact that a trip could not be booked due to scheduling conflicts or eligibility problems).

In addition, the interactive approach (as the name implies), allows the operator to interact with the customer to resolve scheduling conflicts, eligibility questions, out-of-date client data and the like. The problem in achieving successful interactive functioning is the need for processing speed. To put it quite bluntly, customers have better things to do with their time than wait on the telephone while an operator hunts-out information on a computer.

Of the three potentially interactive functions, the least important is updating the master client file. Clearly, this can be performed later at minimum cost to operating efficiency. We feel that not to perform the other functions interactively is not to take advantage of automating the scheduling function. For that reason, the customizing example in Appendix C focuses on these three functions. By customizing these applications, you will be able to

perform them far more quickly than using the standard R:base commands as described in Chapter 2.0. For example, in Chapter 2.0 we recommend printing-out a hard copy of the client master file for reference in checking client eligibility. This can be an inefficient technique in a system with a large number of clients. Customizing this application will enable you to perform this function fast enough to effectively automate it.

3.1.2. File Attributes

Defined below are a number of alternative and additional attribute (field) definitions which the user should consider in developing relations (files). Included are choosing a key field, and selecting and defining client, trip and vehicle specific fields.

3.1.2.1 Choice of Key Fields

As discussed in Chapter 2.0, the selection of key fields in R:base is not crucial because data can be accessed by all fields. The only trade-off is between processing speed and disk storage space. However, many other data base managers require the designation of primary and secondary key fields for accessing data. Therefore, this is an important consideration for users in beginning to automate their systems.

Clearly, you will want to access data by means of a reference to the client as a unique entity. In Chapter 2.0, we assigned each client an ID number. For all but the smallest systems, we recommend the use of ID numbers as the prime means of accessing individual client data. Unlike a last name, you can guarantee that it will be unique.

If you do use a client ID, we also recommend that you continue to use last name as at least a secondary key. Many clients will not be able to provide an ID number on the telephone. In such cases, you will have to revert to access by last name.

Other fields which you might want to consider as keys include city/town/county; human service program numbers; and trip purpose, pay code or passenger classification. The selection of key fields is dependent on how you perceive your data needs particularly in the area of cost allocation/billing and performance evaluation. For example, if you bill human service programs and political jurisdictions by ridership levels, you may want to rapidly access data by town of origin or the client's human service program number. Similarly, if knowing the number of elderly riders is important for evaluating the success of your program, you may want to access data by passenger classification.

3.1.2.2 Client Specific Fields

There are a number of options which the user might want to consider in designing the master client file. For example, to economize on disk storage space you might include only the client's first initial instead of full first name. We do think that providing operators with client first names permits a more personalized approach. Also, with a large client base, you may transport several members of one family. Using initials can cause considerable confusion.

For client address, we collected only street address since Cape Cod, Massachusetts, is composed primarily of single family homes. If your service area has a large number of apartment complexes, we recommend including a field for apartment number. For political jurisdiction, we included "city" because this is the basic unit of government on Cape Cod. You may prefer to substitute or add county and/or neighborhood. If you provide service across state lines, be certain to include a state field. Similarly, if you provide service to more than one telephone area code, you will need a larger telephone field than we included.

We included a "birthdate" rather than "age" field because it requires less frequent update. An age field requires annual update so that you can track when clients cross the elderly threshold. A birthdate field simply needs to be checked each year.

Depending on your data collection requirements, you may want to add an "unduplicated trip" field to the master client file. Many programs are required to track the number of unique riders they carry each month. In other words, while a system may make 1,000 trips per month, only 100 people may account for those trips. As set-up in Chapter 2.0, the only way to count unduplicated passenger trips would be to print-out all trips taken in the month and sort by client ID. You would then have to go through the list manually and count unduplicated riders. In a larger system, this could be tedious.

An alternative is to create a one-column field in the master client file. This field would have only a single possible data entry, either a "y" (yes), "1", or "x". Each month, the appropriate value will be entered into each client's record the first time they book a trip. For each subsequent call, the operator can check to see whether an entry has been made in this field (has the client already used the system this month?). At the end of the month, you would simply need to select those records which contained the value to obtain a list of unduplicated riders. The list could be added manually or you could add a "counter" function similar to that included in some of the reports in Section 2.4. This would add the trips automatically.

As a final step, you will need to delete all the entries to start fresh during the next scheduling period (month or whatever). If you book trips over more than one month simultaneously, you will need to use more than one coding symbol (i.e., 1 = January, 2 = February).

3.1.2.3 Trip Specific Fields

There are a number of options available to the user in designing the vehicle schedule files (prsched, realtime, etc.). As discussed in Section 2.3.3, you may want to include a "trip day" code in the preschedule dummy file to assist in projecting out date-specific preschedule files. The codes should relate to whichever grouping method you have selected (daily, MWF, etc.).

Other simple code fields can be added to meet specific data requirements. For example, suppose you wanted to measure unmet demand among realtime trip requests (people whose travel needs cannot be served). This could be accomplished through the addition of a one-column field to the realtime file.

When a caller requests a trip, the operator would automatically code all the appropriate data fields and then add either a "y" (yes) or "n" (no) to the new field to indicate whether the trip was actually booked or not. At the end of the month, you would select those records where the value equals "y" to measure actual ridership, and those with a value of "n" to measure unmet demand. Since the unmet demand category will contain all the relevant trip data as if the trip had actually been booked, you can disaggregate unmet demand by whatever criteria interest you (i.e., town of origin, destination, age, customer).

Another data category which you might want to evaluate is "no-shows" (people who book a trip and then don't show up for it). This might be of interest for both realtime and prescheduled riders. Again, a one-column field could be created to house a single value indicating "no-show". This value would, of course, have to be entered after-the-fact of the trip when the drivers hand in daily or weekly logs.

An issue which we considered briefly in Chapter 2.0 is the coding of trips for which a client has multiple program eligibility. To handle this problem, we created the "paycode" field. Thus, payment was determined not by whether the client had a medicaid or Title XX number (or both), but by the specific value in "paycode". This problem could also be handled by a priority scheme. For example, local funding rules might hold that medicaid always take priority over Title XX. Thus, a client eligible under both would be funded by Medicaid. If the priority is not absolute but related to trip purpose, the trip purpose field could be used in conjunction with this priority method to determine funding. Either of these methods would eliminate the need for a separate "paycode" field.

A variation on this problem is a trip which has several segments funded by different programs. This would be difficult to handle in the system we established in Chapter 2.0 since we combined both legs of a round-trip into a single record. As discussed in Chapter 2.0, this method has the advantages of reducing data entry time and disk storage requirements. The disadvantage is its lack of responsiveness to non-symmetrical trip patterns. For example, a three corner trip where a rider starts at home and goes to the doctor, the hairdresser, and then home. Medicaid might be willing to pay for only the first leg of the trip.

If you have frequent trips of this type, you should consider three options. You may want to drop the single round-trip booking method and simply book each leg of a trip separately. While this can be more time consuming and increase your storage space requirements, it may be more responsiveness to your needs.

Another option is to add a third-leg to the single round-trip record. In other words, instead of setting up records with round-trip fields as we did, create fields which correspond to "trip leg 1", "trip leg 2", and "trip leg 3". This method will eliminate the need to reenter the client specific data each time. For each leg, include the trip-specific fields which are required for your operation. In particular, if you are apt to run into multiple funding source complications, include a separate "paycode" field for <u>each trip leg</u>.

A third option is to include at least one "return destination field". Because we included only a single destination field, we pretty much limited ourselves to the standard "home-destination-home" trip model. Including additional destination fields would enable you to code the A to B to C type of trip. This would not, however, resolve the problem of assigning each trip leg to a different funding source.

Another field which you might want to consider adding, no matter how you structure your trip schedule file, is drop-off time. Riders may need to know approximate drop-off times and so including this field can be helpful in providing more responsive customer service.

3.1.2.4 Vehicle Specific Fields

The automation of vehicle operations data ("vehops" relation) was not a major feature of Chapter 2.0. Additional data fields which you might want to consider include the following:

- Purchase order number
- Repair order number
- Next scheduled maintenance mileage
- Next scheduled maintenance level

The goal in the expansion of this file is to develop it into a truly active preventative maintenance data base. A major weakness in keeping manual vehicle records, no matter how well done, is retrieving and manipulating the data for the development of an effective preventative maintenance program. Through the addition of fields such as those listed above, and the data manipulation capabilities of programs such as R:base, it will be possible to develop a maintenance experience record for a fleet of vehicles. This will enable you to anticipate and prevent significant maintenance problems.

3.1.3 Reporting Requirements

Section 2.4 describes the process of report generation in great detail and provides some common examples of the types of reports you will likely want to develop. Of course, the options here are literally infinite. In designing your reporting requirements, you should consider the following questions:

- What are the goals and objectives of your system? When you understand what you are trying to accomplish, you can better determine what you need to measure and report on.

- What is the function of your reports? For example, most systems will, at various times, need to produce reports for one of three reasons: 1) provide billing and/or performance data to funding sources; 2) conduct an internal evaluation of system operations; and 3) develop political, institutional and community support for the continued operation/expansion of the system. In designing your reports, consider their ultimate purpose.

- What are the operating characteristics of your system? You want to develop reports which are responsive to the unique design of your system.

- Are you interested in evaluating effectiveness or efficiency? Effectiveness measures how well you are meeting the goals and objectives of your system. For example, if your goal is to service the elderly of your community, an effectiveness measurement would be the number of elderly riders. Efficiency measures how well (i.e., cost-effectively) you produce the service. You could be meeting your goals for elderly ridership at twice what it should be costing you.

Simply by using the fields created in Chapter 2.0 and the select/sort capabilities of a program such as R:base, you can generate the following types of reports:

- Client activity by age, sex, classification, town of origin, funding program, and trip purpose.

- Vehicle utilization by time (compare the hourly driver records in "vehops" to total vehicle availability).

- Any of the following performance measurements:

  - Total vehicle miles
  - Total vehicle hours
  - Subsidy/passenger trip
  - Trips/capita
  - Revenue/passenger
  - Revenue/hour
  - Cost/hour
  - Trips/vehicle
  - Trips (by category)/vehicle
  - Trips/day/vehicle
  - Trips/hour/vehicle

You might also want to consider mileage-based performance measurements such as passenger mile/vehicle or per vehicle mile; or vehicle miles/trip/(vehicle). To do this, you need to create a mileage "look-up" table which ideally an operator can use interactively while booking a trip. This table should be a matrix of the type commonly found on route maps showing distances between frequently used origins and destinations. In our case study, we would probably use the various towns on Cape Cod for this purpose. You could also use counties, neighborhoods, or activity centers. When the operator books a trip, he/she would look-up the mileage on the table and enter it into a field assigned for that purpose. Since mileages are static, it is probably easiest to have the operator look it up from a permanent hard-copy print-out, rather than calling the file up on the screen (although the latter can be done using the customizing features described in Appendix C). After awhile, the operators will tend to memorize the data anyway. You might also want to assign codes to common O-D pairs (i.e., Hyannis-Falmouth = 1). These codes can then be converted to actual mileage figures in the report generation process.

This feature can also be used for billing purposes. In fact, Call-A-Ride and its successor did at times substitute a mileage-based fare for a flat fare due to the tremendously large service area of Cape Cod. Instead of calculating

client, program, and town bills simply by adding the number of trips, they could also be calculated using the mileage involved in the trips. In generating your reports, you can instruct the program to sum the mileage totals in the same way that we summed total trips in Chapter 2.0. You can then apply a per mile dollar value and instruct the program to multiply this value times the total mileage for billing purposes.

3.2 OTHER DATA BASE APPLICATIONS

In this Manual, we concentrated on a limited number of applications for data base management software; scheduling; maintaining a master client file for recordkeeping and eligibility verification; vehicle maintenance recordkeeping; and report generation for billing and performance measurement purposes. There are many other uses for software of this type in paratransit management. In selecting a software product, the user should carefully consider the uses to which it will be put and select a product with the appropriate capabilities and strengths. A sampling of other functions are described briefly below.

3.2.1 Financial Management

Spreadsheets for program budgeting and monthly program financial reports were developed in Chapter 2.0. However, the formal systems for accounting (e.g., Cash Receipts Journal, Cash Disbursements Journal, and General Ledger) are good candidates for customized development with a relational data base and/or integrated spreadsheet.

While there are a number of commercially available microcomputer software packages for home accounting or small business, they are not particularly adaptable to small transit operations. In the former case, they are not sophisticated enough. In the latter, the private-for-profit accounting requirements for tax paying corporations are not proving particularly compatible with public agency and public charity's financial management systems.

For example, these packages are often designed to facilitate income tax preparation, an orientation which will be of little use to the public agency or private-non-profit which pays no income tax. On the other hand, the capability to account for a variety of funding sources and allocate costs accordingly (a crucial feature for the paratransit operator) may be downplayed. These packages, like all software, are rapidly evolving and being adapted to different uses, so the user is advised to compare their capabilities to that of a multi-functional program like R:base.

3.2.2 Payroll

While technically payroll can be considered a component of the financial management system, in the labor-intensive transit business, it has data management significance on its own. In many instances, private non-profit providers are dealing with the Social Security System since January 1, 1984, for the first time.

Dynamic look-up tables and computed fields within records and reports allow significant time saving for transit properties. Calculation of gross pay for individuals based on hours worked, job classification, and hourly rate of pay (the last two using look-up tables) are classic data base management applications. In addition, subtractions from gross pay can be accomplished quite efficiently. For example, Federal, state, and social security taxes can be calculated and withheld using look-up tables. Locally, calculation of payroll deductions for pension or tax sheltered plans, union dues or benefits, Christmas Clubs and other employee-generated deductions, garnishments or court-ordered payroll attachments can be accommodated with a computerized payroll system.

Again, users should also investigate the potential of commercial payroll packages. They need to be sufficiently flexible and sophisticated to meet the unique requirements of government agencies/PNP's (for example, employee deductions may be different than for a private sector employer), and capable of integration with other features of your data base. For example, we built our vehicle performance records from driver's logs which included work hours. You would most likely want to avoid isolating a data item like "work hours" on a separate payroll program and be unable to use it for other data base applications.

3.2.3 Personnel Files

The personnel file is closely related to the Payroll File and would work well together in a relational data base management system. Initially, the personnel file acts as a master file for each employee employed by the organization, and keeps basic information, such as: name of employee, address, telephone number, date of employment, IRS W-4, salary information, EEO and Affirmative Action base data, evaluation dates and history of actions, and date of termination. With a relational data base interacting with the Payroll File, accrual and use of Sick Leave, Annual Leave, Compensatory Time, or Personal Leave are possible. Financial information for the individual, such as year-to-date summaries on wages, taxes withheld, or special payroll deductions are readily available.

Increasingly, personnel information is a major long-term responsibility of the corporation or organization, with increasing legal implications for the employer and employee. This is also an area that gets put off until there is a crisis or litigation.

3.2.4 Fixed Asset Inventory

As paratransit operations mature, many programs are taking over preventative maintenance and, to some degree, fleet repair, rather than contracting out for these services. Inevitably, this means acquiring equipment and tools, as well as consumable supplies. For recordkeeping purposes, these activities are treated differently in tax-paying entities than they are in public sector institutions. In part, this is due to the divergent views that the Internal Revenue Service has of the concept of "capital asset" and the Urban Mass Transportation Administration's determination of an item as "capital equipment" eligible for "capital assistance". Since most commercial inventory packages are written to satisfy the IRS requirements of tax paying sole

proprietorships or corporations, this could be an important area for development on a data base management system, particularly since this is a prime area for potential employee or contractor theft.

3.2.5 Complaints, Incidents, and Accidents

A variety of "other" information which a paratransit agency is usually required to keep can be easily automated. These include complaints, accidents, and other "incidents". The latter might include disputes between employees and management or between employees and customers; medical emergencies; traffic violations; weather-related problems and the like. Using Chapter 2.0 as a model, files can be structured to enable the user to select and sort data in a useful and convenient manner.

APPENDIX A
SELECTING A DATA BASE MANAGER

The purpose of this Appendix is to provide the user with a guide to conducting a selection process for data base management software. Topics include selection criteria; software packages available; characteristics of a good DBMS; and procedures for reviewing products.

A-1 PRIORITIES AMONG THE SELECTION CRITERIA

Based on the needs of a typical paratransit agency, we have prioritized the several criteria involved in the selection of a DBMS as follows:

- Ease of Use is considered to be the most important criteria, assuming that the DBMS possesses the minimum functional criteria. All DBMS packages require a minimum of several hours intensive work to understand how they operate. We also assume a prior understanding of record keeping systems, i.e., their design and operation.

- Capabilities of the DBMS, i.e., its type, size and functions that it can perform. The DBMS should meet all the information needs of the paratransit agency in terms of data storage, organization and retrieval in an efficient manner (e.g., with help screens, menus, and preformatted reports).

- Other Characteristics, such as:

  - integration with other software (spreadsheet, word processors);
  - transferability between machines;
  - upgrade capabilities, i.e., floppy to hard disk storage; single to multi-user environment;
  - quality and readability of the documentation;
  - customer support;
  - manufacturer's reliability.

- Cost of the package - this criterion is of considerable importance given the cost of associated hardware and software.

In summary, the DBMS should be easy to use by the agency staff, and have the minimum threshold of capacity to meet its needs efficiently and at a reasonable cost.

A-2 SOFTWARE AVAILABLE

More than 50 DBMS programs are offered in the microcomputer software market today, and new ones appear each week. Figure A-1 lists nine of the most popular programs. This is a representative set of the different types of DBMS programs offered in the market.

PRODUCT	MANUFACTURER
Data Base Manager II	Alpha Software
Advanced DB Master	Stoneware, Inc.
R:base 4000	Microrim, Inc.
T.I.M. IV	Innovative Software
KnowledgeMan	Micro Data Base Systems
Open Access	Software Products International
Condor 3	Condor Computer Corporation
Probase	Data Technology Industries
Sensible Solution	O'Hanlon Computer Systems
dBase II	Ashton-Tate

FIGURE A-1. POPULAR DBMSs

A-3. REVIEW PROCEDURES

A thorough review of software should consist of three elements in the following order: 1) literature search, 2) personal contact, and 3) field testing. Through the literature search and personal contact, you should be able to narrow the choices to a small number of likely programs.

There are two primary sources of written reviews and guidelines:

- Articles about DBMS concepts and evaluation in several computer magazines such as "BYTE", "InfoWorld", "Popular Computing", "Personal Computing" and "Interface Age", among others.

- Government publications such as: "Microcomputers in Transportation: Selecting a Single User System", "Data Base Management Systems".

Personal contacts should be made with the staffs of paratransit agencies, university professors, and USDOT officials who have had direct experience with specific DBMS products.

Based on these contacts and reviews, manufacturers can be contacted and asked for a review of their product, using either the entire package, a "demo" version or, in its absence, a complete set of specifications. Many companies will send the entire package on a 30-day free trial offer. Demo packages usually perform most system functions at minimal (under $50) cost. You should be extremely reluctant to select a product which is not available for hands-on review. The software industry is in general characterized by manufacturer claims unsubstantiated by field experience and program "bugs" which only become apparent in the field.

A-4 BACKGROUND ON DBMS CONCEPTS

A DBMS can be defined as a set of computer programs that provide data definition, input, storage, retrieval, sorting and manipulation in a useful and efficient manner. In other words, it is a tool for managing an information resource.

The key elements in a data base are the following:

- A field is the simplest unit in the data base. One item of information is stored in each field (e.g., name of a client). Fields have a defined length -- the number of bytes of information contained in the field (e.g., a name field might contain 20 spaces, while a sex field could only have 1). Fields also have a type, such as alphanumeric (combination of letters, numbers and symbols) and numeric (only numbers).

- A record is a collection of fields related to each other (e.g., client name + address + city + state + zip).

- A file is a grouping of records with the same field structure (e.g., client file with name, address, etc. of all the clients).

Thus, a data base is the collection of all the files that have a meaningful relationship to each other.

Three different types of DBMSs exist in the microcomputer environment, according to the levels of sophistication of the data base:

- A List Manager is the simplest DBMS. It is nothing more than a "card" filer program (each "card" holds only one record) that updates, sorts and retrieves the information stored in each "card" separately. In summary, it is something like an electronic "Rolodex".

- A File Management System is capable of working with only one file (comprising several records) at a time. When data is modified in one file, associated data in other files is not changed (i.e., the files are NOT "related" to each other). Being menu driven, it is easy to use, but still limited in capabilities. However, it has certain advantages over a List Manager, such as: the speed of access to the data, data manipulation capabilities using primary or secondary keys (a key is a specially "marked" field used as the base for sorting data), and powerful report generation capabilities.

- A relational DBMS can operate on two or more data files at the same time. The fields in one file are related to the fields in other files. When the data in one of these fields is changed, data in the related fields changes automatically, thus allowing the simultaneous update to numerous pieces of information. This implies the elimination of data redundancy, since fields have to be created only once.

A-5 CHARACTERISTICS OF A GOOD DBMS

A good DBMS should have the following characteristics:

- easy and accurate input of data

- data protection (e.g., user password to access the DBMS; fields are read/write protected)
- data validation ("checking" for input of erroneous data)
- elimination of data redundancy
- quick search and easy update of the data
- report and form generation capabilities
- easy transfer of data to other programs and environments
- availability/manufacturer support

Unfortunately, only a few DBMSs offer all these characteristics, and they are usually difficult to learn and use. There is a clear trade-off between the "power" of a DBMS and its ease of use.

A-6 SELECTION CRITERIA

The following critera should be used in the selection of a DBMS:

- Ease of Use, as previously stated, is the single most important characteristic in a DBMS, recognizing, however, the above mentioned trade-off between "power" and ease of use. Along with the considerations discussed in previous paragraphs, it is necessary to add that some DBMSs are, in reality, designed to be used by a programmer as tools to create finished applications programs. They are essentially code generators that are meant to be used by the skillful user to create customized programs with specific data requirements.

- DBMS's Functional Capabilities (the power of the DBMS) as measured by:

  - the characteristics (as mentioned in Section A-5) that it offers;
  - its "size" in terms of number of fields/record, number of records/file, number of files simultaneously open, etc.;
  - its type (relational or not);
  - data field types that it can handle.

- Integration is one of the "hottest" topics in the microcomputer software field these days. Integrated software means software that provides a high level of functionality and flexibility, passing information effortlessly from one application program to the next, where user data will be stored, shared and retrieved efficiently. The complete MIS would integrate a DBMS, spreadsheet, word processor and graphics program, with the DBMS acting as a starting point -- all current information being entered into it.

  Total integration as described above, however, is almost non-existent today in available commercial microcomputer software. Only one package of which we are aware ("Open Access") claims to integrate all these programs. Open Access is just now becoming available in the market-

place. The other DBMS packages only provided a low level of integration, consisting of the ability to read and/or write data files that can be accessed by the other programs (spreadsheets, word processor).

- Transferability means that the software should be written for a recognized industry-standard operating system (e.g., MS-DOS, CP/M, UCSD-p), compatible with the one used by the spreadsheet and word processing programs, and compatible between different versions of the operating system (if possible).

- Cost - although a DBMS can satisfy the information needs of a paratransit agency more effectively and efficiently than manual methods, the cost of a DBMS program should not be prohibitive. Therefore, a cost ceiling for software procurement must be imposed which is realistic (taking into account the cost of the hardware and other software also needed). The average cost of the packages discussed here is near $500, with a minimum of $295 and a maximum of $695.

A-7 CHARACTERISTICS OF SPECIFIC DBMSs

The characteristics of the DBMSs mentioned at the outset are displayed in Table A-1, showing for each the following information:

- Name and manufacturer.

- Type: relational or file manager (not relational).

- "Size" of the data base, expressed as number of:

  - records/file (number of records in a single file);
  - fields/record (number of "pieces" of information that a record can handle);
  - characters/field (i.e., the maximum length of each field)

- The maximum number of different files simultaneously open. This is an important concept, because if only one file can be open at any time, quite a bit of "shuffling" will be necessary to update, retrieve or enter different records.

- The different data field types that the data base can accept (some accept special field formats such as Social Security number, dollars, date, etc.).

- The level of integration of the DBMS with other programs (spreadsheet, word processor) as defined in the above paragraphs.

- The ease of use is, to some extent, a subjective judgement. We have tried to classify the different programs from a non-technical user's perspective, assuming, however, that the user is willing to spend more than a couple of hours studying the user's manual and tutorial. In this column:

TABLE A-1. DBMSs CHARACTERISTICS

NAME (MANUFACTURER)	TYPE	REC./ FILE	FIELDS/ REC.	CHAR./ FIELD	OPEN FILES	DATA TYPES	DATA PROTECTION	INTEGRATION	EASE OF USE	PRICE
DB MANAGER II (Alpha Software)	File Mgr.	100,000	40	60	1	A,N,D	N/A	Read/Write	Easy	$295
ADVANCED DB MASTER (Stoneware)	File Mgr. (*)	1,000,000 (+)	250	220	1	A,N,D	Yes	Read/Write	Easy	$495
T.I.M. IV (Innoative Soft.)	File Mgr.	32767	40	2400	1/2	A,N,D,$ Other	Password	Read/Write	Easy	$495
R:BASE 4000 (Microrim)	Relational	2.5 billion (+)	400	1530	40	A,N,D,$	Yes	Read/Write	Easy	$495
KNOWLEDGEMAN (MDBS Systems)	Relational	65535	255	65535	Unlim.	A,N,D,L	Yes	Spreadsheet Read/Write	Difficult (Language)	$500
OPEN ACCESS (S.P.I.)	Relational	32000	55	40	5	A,N,D	N/A	Integrated	Easy (+)	$595
CONDOR 3 (Condor)	Relational	65534	127	127	2	A,N,AN,D Other	N/A	Read/Write	Somewhat Difficult	$650
PROBASE (Data Technology)	Relational	65535	64	64	3	A,N,D	N/A	N/A	Difficult (Language)	$650
SENSIBLE SOLUTION (O'Hanlon)	Relational	16 million (+)	384	255	10	A,N,D	Yes	N/A	Difficult	$695
dBase II (Ashton-Tate)	Relational	65535	32	254	2	A,N,L	No	N/A	Difficult (Language)	$700

NOTES: (*) = Sophisticated File Manager
(+) = Claimed by manufacturer

Data Types:
A = Alpha
AN = Alphanumeric
N = Numeric
D = Date
$ = Dollar
L = Logical

- an "easy" program is one that is menu-driven and/or has "help" screens (or lines), with clear documentation that is easy to read and understand;

- a "somewhat difficult" to use program is one that does not provide "help" screens, or is not menu-driven and has more or less obscure documentation;

- a "difficult" to use program is a code-generator type of program (a programming "language" by itself), is hard to learn and is designed to be used by a programmer.

- The current price for each package.

APPENDIX B
USING R:BASE ON FLOPPY DISKS

The key to using R:base on floppy disks is being able to estimate the size of your data base. This process is explained in detail in Section 2.7 of the R:base Manual. The main factors in the size of any data base are the following:

- Number of columns occupied by each record
- Number of records in each relation

There are two other factors unique to R:base. Assigning "keys" to attributes for faster processing adds considerably to the size of the data base. If your data base is small enough to contemplate using floppy disks, the advantages of key fields are minimal. We recommend using no more than one.

R:base also requires temporary holding space for sort files. Once the sort is completed, this space is returned, but you must have enough room to perform the sort.

As you build a data base, you can determine its size through the directory function of your operating system. Each R:base data base consists of three files. In our example, they are labelled as follows:

- CAR1.RBS
- CAR2.RBS
- CAR3.RBS

The first file contains the data base structure and is usually quite small. The second file contains the data and directly relates to the amount of information in your data base. The third file contains key fields. The size of keys will depend on the size of attributes you designate as keys and the amount of data you put in them. You can experiment with the key designation and check its impact and size relative to the data file.

The most important decision you need to make is whether or not to use the "reload" function. As your data base evolves and records and relations are deleted, the space they occupied remains occupied unless you reload the entire data base. To reload, however, you must have available on disk sufficient space for the new, reloaded data base. This data base will equal the size of your original data base minus whatever you deleted. You may decide it is not worth holding space for a reload and just plan to use the entire disk without reloading. If you reload without sufficient space, the entire data base will be killed! Make a back-up copy!

To reload, enter "reload new data base name". Be sure to reload to the "b:" disk unless you have more storage space in the "a:" disk which contains the R:base II diskette. If you do reload to the R:base diskette, you can later use the copy function of your operating system to send the data base back to the working diskette.

After reloading, use the delete command of your operating system to remove the old data base and restore disk space. Then use the rename command to restore the original data base name to the new data base.

The size of a given data base is dependent on the row length of each record and on the number of records. The number of records can be easily estimated. For example, the number of records in the master client file equals the number of clients. The number of records in each of the vehicle schedule files used in this Manual is based on the number of trips assigned to each file. In the system which we set up, round-trips and one-way trips are each contained on a single record. Based on ridership levels and operating procedures, the user should be able to estimate the maximum number of records which will need to be maintained on a given vehicle schedule file at any one time.

To estimate the length of a row, perform the following steps:

1. Add the length of all the attributes in each relation. For this purpose, length is calculated by means of bytes. Attributes have the following byte lengths:

   - Text: number of characters as assigned by user (minimum of 4 bytes)
   - Dollar: 8 bytes
   - Date: 4 bytes
   - Time: 4 bytes
   - Integer: 4 bytes
   - Real: 4 bytes

2. Add 6 bytes per row to the above total. Round odd lengths to an even number.

3. Divide 1536 by the number of bytes in a row for each relation to find the number of rows/block. Discard fractions.

4. Estimate the number of records in each relation.

5. Divide the number of records by the results of Step 3 for each relation to find the number of blocks/relation (round up).

6. Multiply the total number of blocks by 1536 to find the number of bytes in file 2 (b:car2.rbs) of your data base. Compare this amount to available disk space.

R:base also creates temporary files for sorting which are deleted upon completion of the sort. However, there must be sufficient room on disk to perform the sort. To estimate the size of sort files, do the following:

(# of rows being sorted x (sum of length of attributes + 4)) x 2

APPENDIX C
CUSTOMIZING R:BASE

The new features incorporated in R:base (see Appendix F) make it easy to customize the program for specific applications. This is particularly useful when there are a series of commands to be executed repetitively. Operators (such as dispatchers and clerks) who are unfamiliar with R:base commands will be able to perform their particular applications by simply selecting an option from a list of menu choices (previously set up using command files and prompt menus). Thus, the operator does not have to know (or remember) all the commands needed to perform a particular task with R:base. He/she needs only to choose an option from a menu and type whatever the terminal asks for.

Although the new features added to R:base in version 1.1 make it relatively easy to customize, developing customized applications requires a slightly greater degree of data processing sophistication than simply using R:base off-the-shelf. In particular, a rudimentary knowledge of programming in BASIC is helpful in understanding the concepts of command loops, "if" clauses, and other elements of programming logic. Of necessity, the explanation which follows is slightly more technical than the body of the manual.

Almost every step or task in R:base can be conveniently customized. The only exceptions are loading R:base and opening the data base at the beginning of a session. The customization is based on the creation of R:base command files. Every application discussed in the manual includes specific procedures which are executed over and over. Each of these procedures is associated with a given sequence of commands (e.g. selecting a record with specific conditions). Instead of typing the same sequence each time that a given procedure is executed, one can put the commands into an R:base command file and execute the entire sequence by using only the "input filename" command. Thus, the commands are then processed as if they were typed in at the terminal. In other words, the command file is a collection (batch) of R:base commands that are executed simply by typing "input filename" inside R:base ("filename" is the name of the command file).

The utilities that R:base provides to facilitate the creation, editing, listing and running of command files are the R:base commands "rbedit" (a text editor), "type filename" (type and contents of the file at the terminal), and "input filename" (run the command file). An additional feature of command files is that if you put a "~" as the first and only character on a line in the file, R:base will print the message "PRESS ANY KEY TO CONTINUE" at the terminal when that line is encountered during execution of the command file. This feature could be used to pause and step through a series of commands. Command files may also be nested up to five levels deep. This means that you can use a maximum of six command files at any one time (i.e., a maximum of six files can be referenced to each other inside a command file).

The second keystone for the building of a customized application is the use of a new R:base feature, local variables (see Appendix F). These are temporary variables that exist within R:base but are not part of any data base. Each time R:base is loaded, there will be no local variables (i.e., they are "lost" when exiting R:base).

Local variables are created by assigning them a value with "set variable", "fillin" or "compute" commands. (These commands are explained in Appendix F.) Once created, local variables remain active until exiting R:base. Their values can be reset or recomputed at any time by using the same three commands. The command "show variable" will display the names and values of all the current local variables created since the beginning of the session.

Once created, local variables may be used in one of two ways. You can use the value of a local variable within a command by preceding the variable name with a period (.). Secondly, you may use local variables without the period in the conditions of the "if" and "while" commands. (The "if" and "while" commands are used in a command file to conditionally process a series of commands.) These uses (and commands) will be clarified in the following example.

The example that follows demonstrates R:base's customizing capabilities, using the above mentioned commands, local variables and command files. We have to point out, however, that this is only an example of how to customize a specific application. User should build their own customized application(s), suited to meet their specific needs. Nevertheless, this is a useful example of how to build a customized application in which several options can be chosen by the operator, and, at the same time, providing useful insights into the customization process.

C-1. STEP 1 - DEFINE THE APPLICATION TO CUSTOMIZE

The first step should always be to analyze and define the application that you want to customize. In this case, our hypothetical application would be to customize the process of checking the eligibility and updating (if necessary) the information provided by a potential customer who requests service from the paratransit agency. This process is described in Section 2.3.5 of the Manual.

Assume that the operator will use R:base interactively to:

- check if the call that he/she is receiving corresponds to a customer who is listed in the master client file and is eligible for the service requested (e.g., a trip);
- if the client is eligible for the service, add the new trip to the trip file; and,
- update the customer's data, such as change of address, passenger classification, Title XX units remaining, etc.

Also assume that we decide to use a "menu" format to perform each of the above steps, where the operator can choose one of the options to be displayed on the screen. The menu approach greatly reduces the potential for operator error since instead of creating commands from memory as is standard R:base procedure, the operator will respond to computer generated prompts with computer-defined answers. This can also accelerate the process making it more practical to perform these functions while on-line with a customer (hence the term interactive processing).

C-2. STEP 2 - CREATE THE COMMAND FILES AND LOCAL VARIABLES

This step constitutes the "core" of the customization process. Figure C-11 at the end of this Appendix summarizes the steps created in this section.

The first command file to be created is the file called "choices1.cmd". It's displayed in Figure C-1. As can be seen, this file is nothing more than a list of R:base commands such as "newpage", "type", "fillin", etc. The first command, "newpage", is used to clear the screen. Immediately after the screen has been cleared, the command "type a:menu1.dat" displays the contents of the file called "menu1.dat" on the screen. This file is a text file (not a command one) that contains the text of the menu to be displayed on the screen (see Figure C-2).

Once the file "menu1.dat" has been displayed on the screen, the next command line is executed. This is the "fillin" command, used to define the local variable called "msg1" and set its value to the one typed by the operator when he/she makes a choice from the menu displayed (see Appendix F for an explanation of this command).

As shown in Figure C-2, four options are presented to the operator (either "1", "2", "3" or "Q"). The following command lines in Figure C-1 are a series of logical if-endif loops to "branch" to other command files to perform the operation(s) chosen accordingly. For instance, if the operator chooses "1", then the value of "msg1" is set to 1, so the command file called "check.cmd" is input and executed (this file contains the commands to perform the checking of the customer data). The operator could have chosen "Q" instead, executing the "newpage" and "quit" commands and leaving the customized application. ("2" and "3" should be chosen after having performed step "1", as explained below.)

Assuming that the operator chooses option number 1, the command file "check.cmd" (see Figure C-3) will be executed. The first two commands on this file are used to set the null value to "N/A" and to clear the screen with a "newpage" command. Then, a new local variable, "name", is defined, and its value set to the last name of the customer calling, by means of the "fillin" command. The message **"PLEASE TYPE THE LAST NAME OF THE CUSTOMER:"** is displayed on the screen and the program waits for the operator to type that last name, thus setting the value of "name" accordingly. After this, a "newpage" command is executed to clear the screen again.

The following command instructs the program to display the customer's data on the screen, using the "select" command, in order for the operator to verify that the customer is eligible for the trip. Notice that this is very easy to perform with the condition imposed: "where lastname = .name" (this is an example of using local variables in a command line). R:base "knows" that the value of "name" is set to the last name of the customer, so the program searches all the records in the master client file for one with the "lastname" field equal to the value of "name". After displaying the customer's data on the screen, the "output" and "unload" commands are used to write an ascii data file containing the customer's data. The purpose of this is to "store" this useful information for later manipulation and to add this data to the realtime trip file without typing the customer's data again.

```
newpage
type a:menu1.dat
fillin msg1 using "PLEASE CHOOSE ONE OF THE ABOVE (1, 2, 3 or Q):  " +
at 16 8
if msg1 eq 1 then
input a:check.cmd
endif
if msg1 eq 2 then
input a:choices2.cmd
endif
if msg1 eq 3 then
input a:updat2.cmd
endif
if msg1 eq Q then
newpage
quit
endif
quit
newpage
```

FIGURE C-1. COMMAND FILE "CHOICES1.CMD"

```
PLEASE TYPE ONE OF THE FOLLOWING NUMBERS (OR LETTER) TO:

1 - Check if the customer is already listed in the master client file and
    then add the new trip to the trip file;

2 - Update the master client file and related files using a menu;
    (this sould be performed AFTER step 1)

3 - Update the master client file and related files using a screen form;
    (to be performed AFTER step 1)

Q - Quit this menu.
```

FIGURE C-2. TEXT FILE "MENU1.DAT"

```
set null N/A
newpage
fillin name using "PLEASE TYPE THE LAST NAME OF THE CUSTOMER: " +
at 5 10
newpage
select clientid lastname med# xx# xxunits termdate termwhy +
from msclient where lastname = .name
output a:trip.dat
unload data for msclient as ascii using clientid lastname frstname +
address city paxclass specneed where lastname = .name
output terminal
fillin yesno using "DO YOU WANT TO CONTINUE AND ADD THIS NEW TRIP (Y/N)? :" +
at 10 10
if yesno eq Y then
input a:add.cmd
endif
input a:choices1.cmd
quit
```

FIGURE C-3. COMMAND FILE "CHECK.CMD"

The "fillin" command is used again to set the new local variable "yes/no" to the value typed by the operator as an answer to the question displayed: **DO YOU WANT TO CONTINUE AND ADD THIS NEW TRIP (Y/N)?.** If the operator, upon examination of the customer's data, decides that the customer is not eligible for the trip, the answer would be "N". The command "input a:choices1.cmd" will be executed, displaying again the file "menu1.dat", without adding that trip to the realtime trip file. Otherwise (answer "Y"), the command "input a:add.cmd" will be performed and the file "add.cmd" executed.

The command file "add.cmd" (see Figure C-4) is used to add the customer's data to the realtime trip file. It begins clearing the screen ("newpage") and displaying the contents of the textfile "addtxt.dat" with the "type a:addtxt.dat" command. This file (see Figure C-5) contains the message to be displayed on the screen to inform the operator what to do next: to use the form to be displayed to fill the remaining data of the customer's trip characteristics (time, destination, etc.) to the realtime trip file. While this message is displayed, the customer's data previously stored in the data file "trip.dat" is being loaded to the "realtime" trip file by means of the "load" command, thus automatically filling in those fields without the operator having to retype them.

Thus, data common to both the master client and vehicle schedule files (name, address, classification, etc.) are loaded automatically. As you recall, in the example in the text, each attribute had to be loaded by the operator individually.

The "set rules off" and "set messages off" commands are necessary in this example to override the error message which we had created in the establishment of the data base. As you recall, we had created a rule not to allow a trip to be scheduled before 7:00 a.m. When the client file data is loaded automatically, no trip times have yet been established. R:base reads trip time as having a "null" value which it interprets as preceding 7:00 a.m. Thus, unless rules are turned off, our error message will be triggered. This reaction frankly surprised us. Since we had not created a rule requiring a pick-up time, we had assumed that R:base would simply ignore the null value. After the loading has been completed, both rules and error messages are again "turned on".

The "~" character is used to display on the screen the message: **PRESS ANY KEY TO CONTINUE.** The screen then clears and the "rtsched" form (set up with the "realtime" relation) appears in the screen by means of the "edit using rtsched" command, with the customer's data filled in because of the condition "where lastname = .name" (and because of the previous "load" command). The operator then "browses" through the form filling in the trip data and editing, if necessary, the other fields in the form. Note that this can be done interactively while communicating with the customer on the phone at the same time. By pressing the "C" option, the trip data is added to the realtime trip file. Then the command "input a:choices1.cmd" brings up the file "menu1.dat" (i.e., our first menu) to the screen again.

Options "2" and "3" of the original menu provide the operator with the ability to update the customer's data in the master client file due to a change of address, classification, Title XX units, etc. This is accomplished with one

```
newpage
type a:addtxt.dat
set rules off
set messages off
load realtime from a:trip.dat as ascii using clientid lastname frstname +
address city paxclass specneed
set messages on
set rules on
~
set null ---
edit using rtsched where lastname = .name
input a:choices1.cmd
```

FIGURE C-4. COMMAND FILE "ADD.CMD"

```
PLEASE USE THE FOLLOWING FORM

TO FILL IN THE BLANK FIELDS

AND THEN PRESS "C" TO ADD THE NEW TRIP

TO THE REAL TIME TRIP FILE
```

FIGURE C-5. TEXT FILE "ADDTXT.DAT"

of these two choices. Both are presented in this example to demonstrate the various possibilities existing in the setup of a customized application. Option "2" uses a "menu approach" in which the operator chooses the field that has to be updated. Option "3" uses a form to display all the customer's data and allowing the operator to "browse" through it, editing the fields to be changed accordingly. Note that these options should be performed after option "1" since both make use of the local variable "name" set up in option "1". Thus, if the operator chooses "2" or "3" as the first option, the local variable "name" does not exist and the commands that use it have no effect. The advantage of each approach is described below.

If the operator chooses option "2" then the command file "choices2.cmd" (see Figure C-6) is executed (by means of the command "input a:choices2.cmd" in the "choices1.cmd" command file). After clearing the screen and setting the number of lines per screen to 24 (in order to display the following menu), the contents of the text file "menu2.dat" are displayed on the screen (see Figure C-7). The operator then chooses one of the options and sets the value of the new local variable "msg2" with the "fillin msg2 using..." command. Then the command file "update.cmd" is input and executed.

The command file "update.cmd" (see Figure C-8) is used to update (change) the value of the field chosen by the operator. It consists of several if-endif logical loops that are performed depending on the value of the local variable "msg2". For example, if the operator chooses to change the address of the customer, then option "A" of the menu is chosen, thus setting "msg2" equal to A. Then the first logical if-endif loop is performed. The screen is cleared and a new local variable, "field", is defined using the command "fillin field using **"PLEASE ENTER THE NEW ADDRESS:"",** its value equal to the string typed in by the operator as the new address of the customer. Then the field "address" is changed over all relations containing it by means of the command "change address to .field where lastname = .name". Notice the use of the previously set (and still active) local variable "name". Note also the power of the relational data base in permitting simultaneous update of a field across several files.

The updating of the other fields is identical, setting first the value of "field" to the string typed by the operator and then changing the value of the particular field to the "field" value over all relations where the condition "lastname = .name" holds (i.e., where the data corresponds to the customer's data). After changing the field(s), the menu ("menu2.dat") will again be displayed by means of the command "input a:choices2.cmd" in the "update.cmd" command file. The operator can then change other fields or quit this process by choosing "Q", thus leaving the customized application.

Our example ends with the last option ("3") that can be chosen from the first menu ("menu1.dat"). As said before, this option uses a form to change (update) the customer's master file data. When the operator chooses "3" the command file "updat2.cmd" is input (see Figure C-9) and executed (by means of the command "input a:updat2.cmd" in the "choices1.cmd" command file). After the screen is cleared, the text file "message.dat" is displayed (see Figure C-10) on the screen. Then, the "client" form (set up with the master client file "msclient") is used to edit the customer's data using the command "edit using client where lastname = .name".

```
newpage
set lines 24
type a:menu2.dat
fillin msg2 using "PLEASE CHOOSE ONE OF THE ABOVE (A,T,Z,P,C,N,X,D,R or Q): " +
at 24 8
input a:update.cmd
```

FIGURE C-6. COMMAND FILE "CHOICES2.CMD"

```
PLEASE TYPE ONE OF THE LETTERS TO UPDATE THE FOLLOWING
FIELD(S) IN THE MASTER CLIENT FILE AND RELATED FILES:

A - Change the customer's address;

T - Change the customer's city/town of residence;

Z - Change the customer's zip code;

P - Change the customer's phone number;

C - Change the customer's passenger classification;

N - Change the customer's special needs information;

X - Change the customer's Title XX units;

D - Change the customer's termination date;

R - Change the customer's termination reason;

Q - To quit this menu.
```

FIGURE C-7. TEXT FILE "MENU2.DAT"

```
newpage
if msg2 eq A then
fillin field using "PLEASE ENTER THE NEW ADDRESS: " at 5 10
change address to .field where lastname = .name
endif
if msg2 eq T then
fillin field using "PLEASE ENTER THE NEW CITY/TOWN: " at 5 10
change city to .field where lastname = .name
endif
if msg2 eq Z then
fillin field using "PLEASE ENTER THE NEW ZIP: " at 5 10
change zip to .field where lastname = .name
endif
if msg2 eq P then
fillin field using "PLEASE ENTER THE NEW PHONE NUMBER: " at 5 10
change phone to .field where lastname = .name
endif
if msg2 eq C then
fillin field using "PLEASE ENTER THE NEW PASSENGER CLASS: " at 5 10
change paxclass to .field where lastname = .name
endif
if msg2 eq N then
fillin field using "PLEASE ENTER THE NEW SPECIAL NEEDS: " at 5 10
change specneed to .field where lastname = .name
endif
if msg2 eq X then
fillin field using "PLEASE ENTER THE NEW TITLE XX UNITS: " at 5 10
change xxunits to .field where lastname = .name
endif
if msg2 eq D then
fillin field using "PLEASE ENTER THE NEW TERMINATION DATE: " at 5 10
change termdate to .field where lastname = .name
endif
if msg2 eq R then
fillin field using "PLEASE ENTER THE NEW TERMINATION REASON: " at 5 10
change termwhy to .field where lastname = .name
endif
if msg2 eq Q then
quit
endif
input a:choices2.cmd
```

FIGURE C-8. COMMAND FILE "UPDATE.CMD"

```
newpage
type a:message.dat
edit using client where lastname = .name
set variable strt to address in msclient where lastname = .name
set variable town to city in msclient where lastname = .name
set variable post to zip in msclient where lastname = .name
set variable bell to phone in msclient where lastname = .name
set variable class to paxclass in msclient where lastname = .name
set variable need to specneed in msclient where lastname = .name
set variable unit to xxunits in msclient where lastname = .name
set variable tdate to termdate in msclient where lastname = .name
set variable twhy to termwhy in msclient where lastname = .name
change address to .strt where lastname = .name
change city to .town where lastname = .name
change zip to .post where lastname = .name
change phone to .bell where lastname = .name
change paxclass to .class where lastname = .name
change specneed to .need where lastname = .name
change xxunits to .unit where lastname = .name
change termdate to .tdate where lastname = .name
change termwhy to .twhy where lastname = .name
quit
```

FIGURE C-9. COMMAND FILE "UPDAT2.CMD"

```
PLEASE USE THE FOLLOWING FORM

TO CHANGE (EDIT) ANY OF THE

FIELDS OF THE MASTER CLIENT FILE

AND RELATED FILES
```

FIGURE C-10. TEXT FILE "MESSAGE.DAT"

The operator "browses" through the form, editing all the fields to be updated. By pressing the "C" option, the new data will be written to the master client file, "msclient". In this way, only the "msclient" relation will be updated. The other relations containing all or some of the same fields that have been updated will be unchanged. To effectively change the values of the new customer's data in all the other relations the following groups of commands are used in the "updat2.cmd" command file. First, the new local variables "strt", "town", "post", etc., are set equal to the new values (just updated with the "client" form) of the fields "address", "city", "zip", etc., in the "msclient" relation where "lastname = .name" (i.e., for the customer's data). These new local variables are then used to change the values of the "address", "city", "zip", etc., fields in all the other relations by means of the commands "change address to .strt where lastname = .name", etc. Thus, simply by editing the form displayed on the screen for the master client file, all the fields changed are updated over all the relations containing those fields for the specific customer that is calling. Note, again, that this can be done interactively while the operator is communicating with the customer.

The advantage of the menu approach (Option 2) is that the operator can quickly update whichever field of data needs updating. It is most efficient for updating single pieces of data because it eliminates the need to "browse" through the entire "client" form to find the field which requires updating. It also performs a true relational operation since it updates the field in all relations in which it is contained. As discussed in the text, most of the time this is a desirable feature.

The form approach (Option 3) is probably faster for updating several data fields simultaneously. Instead of returning to the menu to call up each field individually as in Option 2, the entire form is displayed for the operator, who can browse through and selectively update. Since each form in R:base is associated with ONE AND ONLY ONE relation (see Section 2.2.6.1), using this approach will not normally result in a relational operation. The data will only be changed in the one relation associated with the form. It is possible to achieve a relational outcome with further manipulation as described above.

There may be occasions, however, where you will not want to update a field across all relations. For example, suppose you bill towns on the basis of trips by residents. If a client moves to a new town after having taken several trips from his/her old address earlier in the month, you will want to bill those trips to the original town of residence. Therefore, while you will want to update the master client file, you will not want to update the master vehicle schedule file which contains all completed trips for the month.

C-3 STEP 3 - REVIEW AND TEST

The final step in the customization process is to review the command and text files that have been created by using the "rbedit" text editor to correct any errors in the commands or text. After that, test the procedure to be sure that it meets your requirements. To do this, execute the command files by using the "input" command. To see the processing of these command files for

test purposes, it is useful to use the "set echo on" command. All commands in the file will be executed in sequence and then control returned to the terminal. If the test does not show the results expected, use the "rbedit" text editor to analyze and correct any errors in the commands.

C-4 SUMMARY

As a summary of this customization example, the following flowchart (see Figure C-11) shows all the steps to be performed for this particular application, i.e., to add a new trip to the real time trip file and update the customer's data interactively.

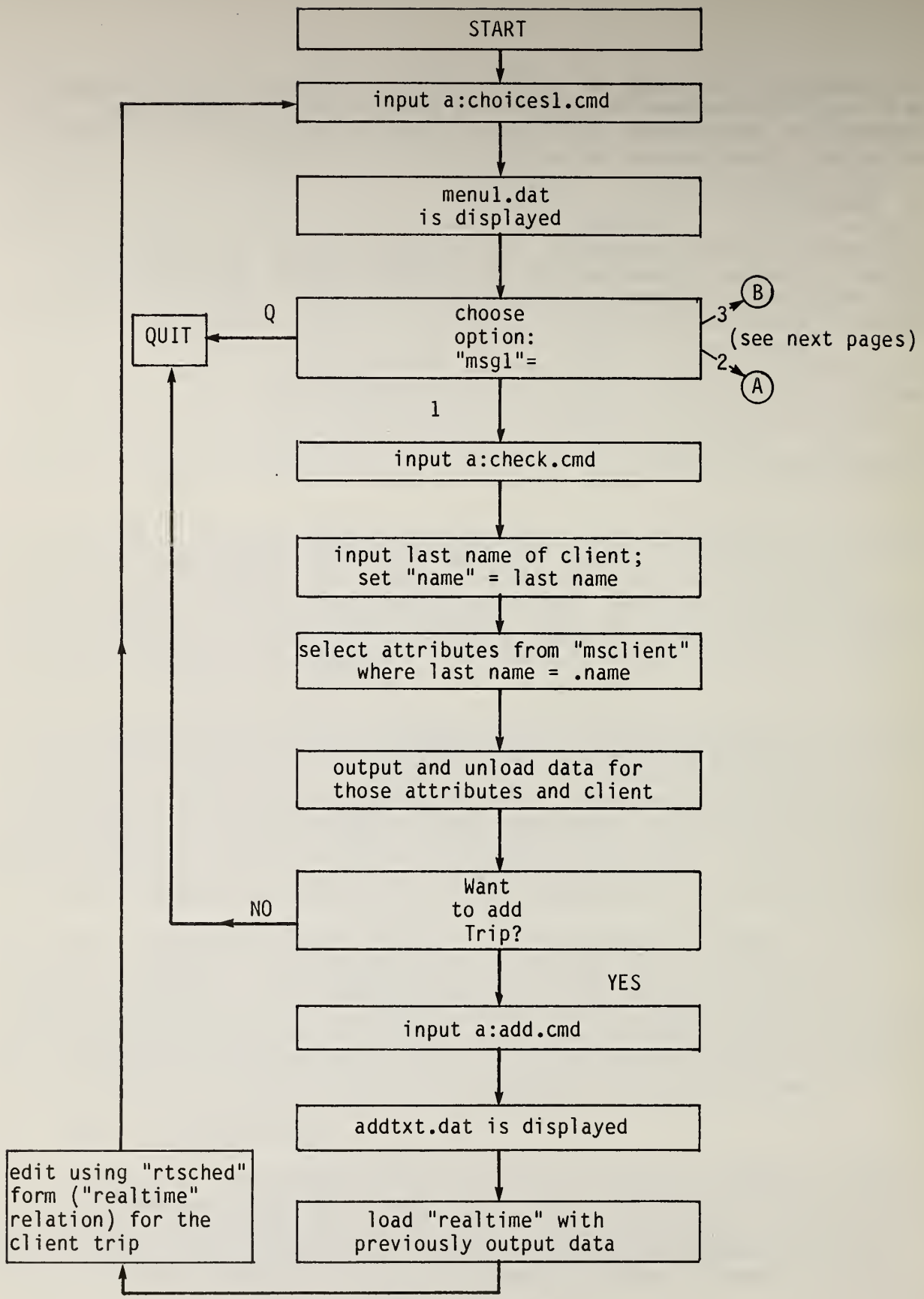


FIGURE C-11. SUMMARY OF CUSTOMIZING EXAMPLE

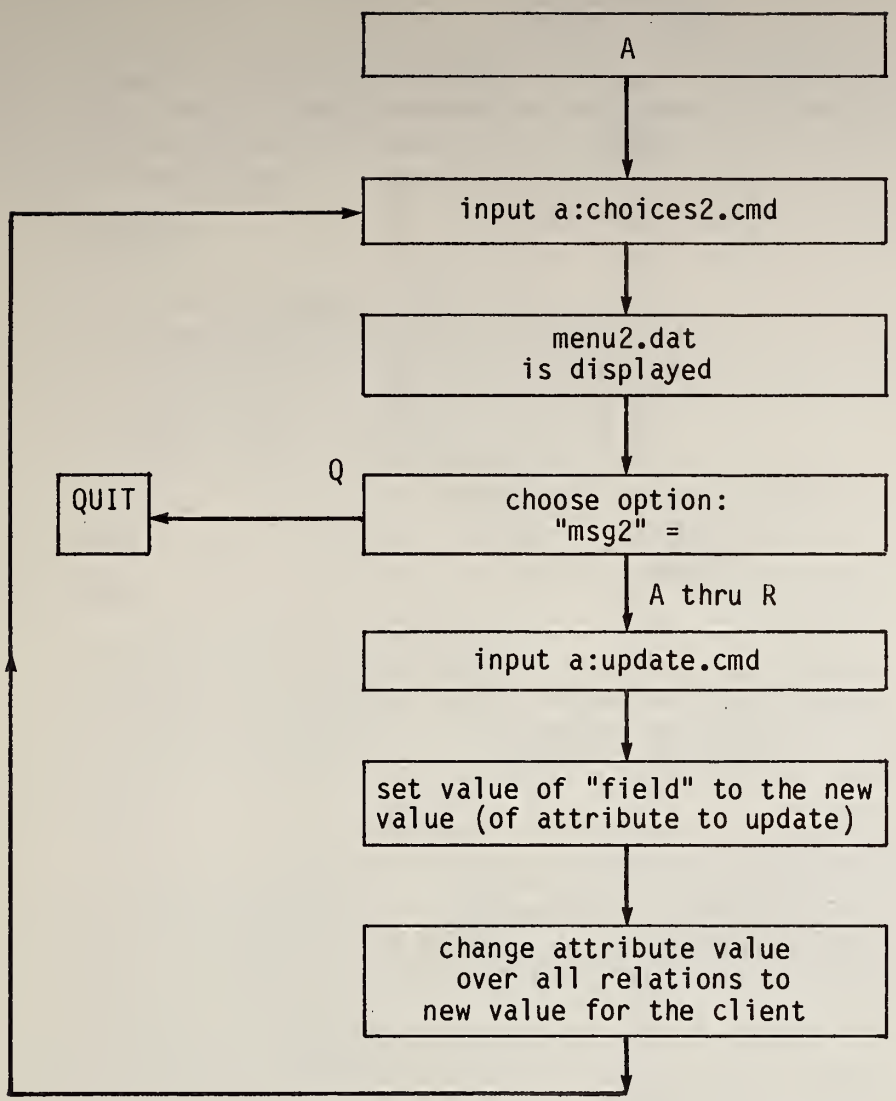


FIGURE C-11. SUMMARY OF CUSTOMIZING EXAMPLE (Continued)

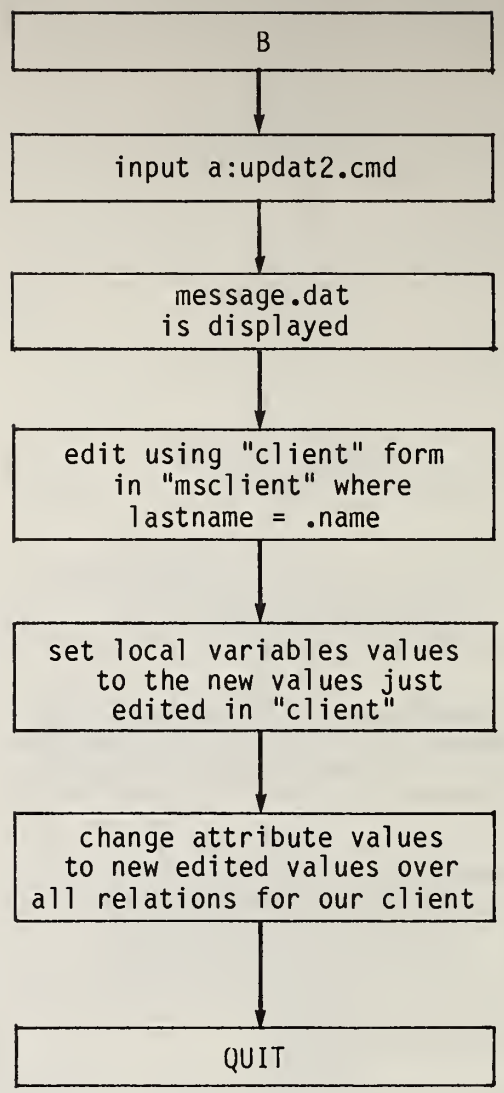


FIGURE C-11. SUMMARY OF CUSTOMIZING EXAMPLE (Concluded)

APPENDIX D
SCREEN FORMS

This Appendix contains all screen forms used to build the case study data base, CAR, which forms the basis of this manual. You can utilize these forms to replicate all of the functions performed in this manual. The following forms are provided:

1. Relations

   - msclient (master client file)
   - prsched (prescheduled trip file)
   - realtime (dial-a-ride trip file)
   - vsched (union of prsched and realtime)
   - vehops (vehicle operations file)
   - vmaster (master vehicle file)

2. Rules

3. Forms

   - client (msclient relation)
   - schedule (prsched)
   - rtsched (realtime)
   - operate (vehops)

4. Reports

   - vehstats (vehops)
   - trips (vsched)
   - clfile (msclient)
   - bill (vsched)
   - citybill (vsched)
   - medbill (vsched)
   - vehsched (vsched)

D-1. RELATIONS

```
Relation: msclient
Read Password: YES
Modify Password: YES

          Attributes
  #  Name       Type      Length           Key
  1  clientid   TEXT        4 characters   yes
  2  lastname   TEXT       15 characters
  3  frstname   TEXT       10 characters
  4  address    TEXT       15 characters
  5  city       TEXT       10 characters
  6  zip        TEXT        5 characters
  7  phone      TEXT        8 characters
  8  paxclass   TEXT        3 characters
  9  sex        TEXT        1 characters
 10  brthdate   DATE        1 value(s)
 11  specneed   TEXT       10 characters
 12  med#       TEXT        6 characters
 13  xx#        TEXT        6 characters
 14  xxunits    TEXT        2 characters
 15  termdate   DATE        1 value(s)
 16  termwhy    TEXT       15 characters

          Attributes
  #  Name       Type      Length           Key
 17  editdate   DATE        1 value(s)

Current number of rows:     141
```

FIGURE D-1. MSCLIENT RELATION

```
Relation: prsched
Read Password: YES
Modify Password: YES

          Attributes
  #  Name      Type      Length          Key
  1  tripdate  DATE        1 value(s)
  2  veh#      TEXT        2 characters
  3  clientid  TEXT        4 characters  yes
  4  lastname  TEXT       15 characters
  5  frstname  TEXT       10 characters
  6  address   TEXT       15 characters
  7  city      TEXT       10 characters
  8  paxclass  TEXT        3 characters
  9  specneed  TEXT       10 characters
 10  putime    INTEGER     1 value(s)
 11  destinat  TEXT       15 characters
 12  rt        TEXT        1 characters
 13  rttime    INTEGER     1 value(s)
 14  trip#     INTEGER     1 value(s)
 15  trippurp  TEXT        3 characters
 16  paycode   TEXT        4 characters

          Attributes
  #  Name      Type      Length          Key
 17  editdate  DATE        1 value(s)

Current number of rows:     62
```

FIGURE D-2. PRSCHED RELATION

```
Relation: realtime
Read Password: YES
Modify Password: YES

          Attributes
  #  Name      Type      Length          Key
  1  tripdate  DATE         1 value(s)
  2  veh#      TEXT         2 characters
  3  clientid  TEXT         4 characters yes
  4  lastname  TEXT        15 characters
  5  frstname  TEXT        10 characters
  6  address   TEXT        15 characters
  7  city      TEXT        10 characters
  8  paxclass  TEXT         3 characters
  9  specneed  TEXT        10 characters
 10  putime    INTEGER      1 value(s)
 11  destinat  TEXT        15 characters
 12  rt        TEXT         1 characters
 13  rttime    INTEGER      1 value(s)
 14  trip#     INTEGER      1 value(s)
 15  trippurp  TEXT         3 characters
 16  paycode   TEXT         4 characters

          Attributes
  #  Name      Type      Length          Key
 17  editdate  DATE         1 value(s)

Current number of rows:      38
```

FIGURE D-3. REALTIME RELATION

```
Relation: vsched
Read Password: YES
Modify Password: YES

          Attributes
  #  Name      Type      Length          Key
  1  tripdate  DATE        1 value(s)
  2  veh#      TEXT        2 characters
  3  clientid  TEXT        4 characters
  4  lastname  TEXT       15 characters
  5  frstname  TEXT       10 characters
  6  putime    INTEGER     1 value(s)
  7  address   TEXT       15 characters
  8  city      TEXT       10 characters
  9  destinat  TEXT       15 characters
 10  trip#     INTEGER     1 value(s)
 11  rt        TEXT        1 characters
 12  rttime    INTEGER     1 value(s)
 13  paxclass  TEXT        3 characters
 14  trippurp  TEXT        3 characters
 15  paycode   TEXT        4 characters
 16  specneed  TEXT       10 characters

          Attributes
  #  Name      Type      Length          Key
 17  editdate  DATE        1 value(s)
 18  med#      TEXT        6 characters
 19  xx#       TEXT        6 characters

Current number of rows:     253
```

FIGURE D-4. VSCHED RELATION

```
Relation: vehops
Read Password: NO
Modify Password: YES

          Attributes
  #  Name       Type       Length          Key
  1  veh#       TEXT          2 characters
  2  opsdate    DATE          1 value(s)
  3  driver#    TEXT          2 characters
  4  endmile    INTEGER       1 value(s)
  5  stmile     INTEGER       1 value(s)
  6  opsmile    REAL          1 value(s)
  7  endhour    INTEGER       1 value(s)
  8  sthour     INTEGER       1 value(s)
  9  opshour    REAL          1 value(s)
 10  fuelqty    REAL          1 value(s)
 11  fuelcost   REAL          1 value(s)
 12  mpg        REAL          1 value(s)
 13  oilqty     INTEGER       1 value(s)
 14  oilcost    REAL          1 value(s)
 15  repair     TEXT         15 characters
 16  repcost    REAL          1 value(s)

          Attributes
  #  Name       Type       Length          Key
 17  editdate   DATE          1 value(s)

Current number of rows:      25
```

FIGURE D-5. VEHOPS RELATION

```
Relation: vmaster
Read Password: NO
Modify Password: YES

          Attributes
  #  Name       Type       Length          Key
  1  veh#       TEXT         2 characters
  2  vehid      TEXT        10 characters
  3  lic#       TEXT         6 characters
  4  yrmfg      TEXT         4 characters
  5  make       TEXT         6 characters
  6  model      TEXT         6 characters
  7  engsz      TEXT         6 characters
  8  title      TEXT        10 characters
  9  lien       TEXT        10 characters
 10  purdate    DATE         1 value(s)
 11  radio      TEXT         1 characters
 12  access     TEXT         1 characters
 13  #wchair    INTEGER      1 value(s)
 14  cap        INTEGER      1 value(s)
 15  GVW        TEXT         4 characters
 16  mipurch    TEXT         5 characters

          Attributes
  #  Name       Type       Length          Key
 17  nowmiles   INTEGER      1 value(s)
 18  comment    TEXT        15 characters
 19  editdate   DATE         1 value(s)

Current number of rows:       5
```

FIGURE D-6. VMASTER RELATION

D-2. RULES

```
     RULE checking = ON
RULE   1     zip ge 0200
         and zip le 02999
    Message:Zip code is incorrect
RULE   2     clientid IN msclient nea clientid IN msclient
    Message:Client ID is a duplicate
RULE   3     paxclass exis
    Message:Missing passenger classification
RULE   6     putime ge 0700
    Message:Trip time is too early!
RULE   7     rt eq y
         or  rt eq n
    Message:Round trip code is incorrect
RULE   8     paxclass eq e
         or  paxclass eq eh
         or  paxclass eq h
         or  paxclass eq weh
         or  paxclass eq wh
    Message:Passenger code is incorrect
RULE   9     trippurp eq hc
         or  trippurp eq nu
         or  trippurp eq mow
         or  trippurp eq se
         or  trippurp eq ft
         or  trippurp eq gt
         or  trippurp eq adc
         or  trippurp eq dl
    Message:Trip code is incorrect
RULE  10     endhour IN vehops gta sthour IN vehops
    Message:End hours must be greater than start hou
RULE  11     endmile IN vehops gta stmile IN vehops
    Message:End miles must be greater than start mil
RULE  12     endmile IN vehops gta stmile IN vehops
    Message:End miles must be greater than start mil
RULE  13     endhour IN vehops gta sthour IN vehops
    Message:End hours must be greater than start hou
```

FIGURE D-7. RULES

D-3. FORMS

```
Client ID  S  E

Last Name   S            E       First Name   S        E
Address     S            E       City  S        E       Zip  S   E
Telephone   S      E

Passenger Class  S E     Sex   E   Birthdate   S      E
Special Needs    S        E

Medicaid #  S    E   Title XX #  S    E    Title XX units   SE
Termination Date  S      E    Reason  S             E

Edit Date  S      E
```

FIGURE D-8. CLIENT FORM

```
Trip Date  S      E         Vehicle #   SE

Client ID S  E   Last Name  S             E      First Name  S        E
Address   S             E      City  S        E
Passenger Class  S E      Special Needs  S        E

Pick-Up Time    S        E      Destination  S             E
Round Trip?     E              Return Time  S        E
Trip #   S        E

Trip Purpose  S E     Payment   S  E

Edit Date  S      E
```

FIGURE D-9. SCHEDULE FORM

```
Trip Date  S      E          Vehicle #   SE

Client ID  S  E    Last Name  S            E    First Name  S        E
Address    S              E       City   S        E
Passenger Class   S E        Special Needs   S        E

Pick-Up Time  S        E        Destination   S             E
Round Trip?   E                 Return Time   S        E
Trip #        S        E

Trip Purpose  S E      Payment  S  E

Edit Date    S      E
```

FIGURE D-10. RTSCHED FORM

```
Vehicle # SE    Ops Date  S      E    Driver #   SE

End Mile  S        E  Start Mile S        E   Ops Miles  S        E
End Hour  S        E  Start Hour S        E   Ops Hours  S        E

Fuel Used  S      E   Cost   S        E        MPG   S        E
Oil Used   S      E   Cost   S        E

Repair  S              E             Cost  S        E

Edit Date  S      E
```

FIGURE D-11. OPERATE FORM

D-4. REPORTS

```
H                                    VEHICLE STATISTICS
H
H  Veh#  Trip Date    Hours     Miles     Fuel     Cost      MPG     Oil     Cost      Repair        Cost
H  ----  ----------  -------   -------   ----     ----      ----    ---     ----      ------        ----
D  SE    S       E   S   E     S    E    S   E    S    E    S   E   E       S   E     S        E    S   E
F
F
F
F
F  Total:           S   E     S    E         S    E    S   E                                        S   E
F
F  Average MPG:                                          S   E
F
F  Miles/Trip:              S    E
F
F  Trips/Mile:              S    E
F
F  Trips/Hour:      S   E
```

FIGURE D-12. VEHSTATS REPORT

```
H
H                                    RIDERSHIP BY CLIENT
H
H     ID    Last Name      First Name      Address          City       Class  Purpose  Pay  Trip Date      Trip#   Edit Date
H     ----  ----------     ----------      -------          ----       -----  -------  ---  ----------     -----   ----------
D     S E   S          E   S       E       S          E     S      E   S E    S E      S E  S      E       E       S      E

F
F     TOTAL TRIPS:  S      E
```

FIGURE D-13. TRIPS REPORT

```
H                                    MASTER CLIENT FILE
H
H
H
H                                                  c                   u
H                                                  l                   n
H                                                  a                   i
H                                                  s                   t
H Last Name    First Name   ID   Address      City     s  SpecNeed  Med#  XX#  s  Term Date  Term Why?  EditDate
H ---------    ----------   --   -------      ----     -  --------  ----  ---  -  ---------  ---------  --------
D S       E S      E S E  S       E S     E S E S    E S   E S   E  SE S    E  S       E S    E
```

FIGURE D-14. CLFILE REPORT

```
H                                   CLIENT BILL
H                                   Month: May
H
H    Name: S            E  S        E ID: S  E
H    Address: S            E City:  S       E
H
H    Trip Date     Destination        Purpose    Trip#    Paycode
H    ---------     -----------        -------    -----    -------
D     S      E     S             E     S E       E        S  E
F
F    TOTAL TRIPS:  S    E
F
F    TOTAL COST:   S    E
F
F    Purpose Code: hc=health care; nu=site nutrition; mow=meals on wheels;
F                  se=special education; ft=field trips; gt=group trips;
F                  adc=adult day care; dl=dialysis
F
F    Trip# equals the number of trips taken on a given day.  If you rode the
F    bus both ways, trip# will equal "2".  If you were accompanied by an
F    escort, the escort's trips will also be counted.
```

FIGURE D-15. BILL REPORT

```
H                                        CITY ASSESSMENT
H                                        City: S       E
H                                        Month: May
H
H       ID    Last Name      First Name    Address          Tripdate    Destination     Trip#  Class  Purpose  Pay
H       ----  ---------      ----------    -------          --------    -----------     -----  -----  -------  ---
D       S E   S          E S       E   S          E  S     E   S           E  E    S E    S E    S E

F
F       TOTAL TRIPS: S    E
F
F       TOTAL COST:  S    E
F
F       Purpose Code: hc = health care; nu = nutrition site; mow = meals on w
F                     ft = field trips; gt = group trips; adc = adult day car
F
F       Passenger Code: e = elderly; h = handicapped; eh = elderly handicappe
F                       weh = wheelchair elderly handicapped.
```

FIGURE D-16. CITYBILL REPORT

```
H                                   MEDICAID INVOICE
H                                   Month: May
H
H     Med#    Last Name       First Name    Trip Date   Destination      Trip#    Pay
H     ------  ---------       ----------    ---------   -----------      -----    ---
D   S     E   S           ES         E   S       E   S            E   E      S  E

F
F   TOTAL TRIPS:   S       E
F
F   COST:          S       E
```

FIGURE D-17. MEDBILL REPORT

```
H                                        VEHICLE SCHEDULE
H                                        Vehicle #: SE
H                                        Date: S     E
H
H      Time  Last Name     First Name   Address        City       Destination    RT Time  Class  Purpose  Special Needs
H      ----  ---------     ----------   -------        ----       -----------    -------  -----  -------  -------------
D      S E   S          E  S        E   S          E   S       E  S           E  S E    S E    S E    S        E
```

FIGURE D-18. VEHSCHED REPORT

D-5. COMPLETE MASTER CLIENT FILE

Last Name	First Name	ID	Address	City	Class	SpecNeed	Med#	XX#	Units	Term Date	Term Why?	EditDate
Abbott	Catherine	0064	9702 Bridge	Yarmouth	e	no phone				05/31/79	moved	05/31/79
Allen	Helen	0146	50 Lawtner	Centervill	eh		628982					05/08/79
Allenson	Herbert	0154	55 Rustown	Cotuit	weh			251753	30	09/30/79		05/14/79
Aries	Rose	0077	15 Fringewood	Dennis	e	medicatio						05/03/79
Armas	Marie	0071	12 Snow	Dennis	e							05/16/79
Auld	Grace	0031	27 Tealford	Dennis	eh		019952	275743	22	08/31/79		05/15/79
Auld	Nellie	0149	108 San Marcus	Falmouth	eh		043904	556433	34	09/30/79		05/11/79
Avallon	Kathleen	0072	1 Vickie	Dennis	e	structure						05/17/79
Baker	John	0001	65 Bass	Dennis	e							05/01/79
Barrett	Robert	0129	66 Kenwood	Falmouth	e		293173					05/11/79
Battis	Edith	0155	481 Oates	Hyannis	weh		986152					05/15/79
Bensten	Ellen	0090	4 Beachway	Hyannis	eh	structure						05/12/79
Berman	Helen	0006	23 Sheryl	Yarmouth	e	medicatio						06/11/79
Biden	Manuel	0150	29 Coronation	Falmouth	eh							05/12/79
Boggs	Vickie	0122	13 Dyer	Pocasset	eh		470145					05/17/79
Boisvert	Mildred	0056	3 Alford	Dennis	e							05/09/79
Boren	Hilda	0148	10 Downing	Falmouth	e					05/31/79	overdue bill	05/31/79
Boyd	Thomas	0160	55 Astor	Sandwich	eh							05/19/79
Bradley	Ruth	0041	5 Ryan	Yarmouth	eh							05/23/79
Buckley	Marguerite	0091	108 Syracuse	Falmouth	e	escort						05/13/79
Buckner	Marion	0042	196 Sandra	Yarmouth	h							05/24/79
Buckner	Eliza	0021	109 Lance	Hyannis	eh		017220	111341	12			05/09/79
Bumpers	Blanche	0143	120 Ocean Ave	Falmouth	eh		022116					05/05/79
Busheuff	Mary	0157	4 Point East	Bourne	en	medicatio	920527					05/17/79
Cabrera	Florence	0162	73 Larry	Bourne	weh		695160					05/21/79
Callahan	Arthur	0049	608 Mercedes	Yarmouth	e							05/04/79
Carmichael	Lillian	0027	19 Fairhaven	Yarmouth	e							05/12/79
Chaffee	Claire	0045	9 Woodwind	Dennis	e		167177	497833	02	05/31/79		05/30/79
Chiles	Bertha	0082	25 Eastwood	Harwich	e							05/05/79
Clark	Paula	0118	1578 Samuell	Falmouth	h		144202					05/20/79
Clear	Alice	0164	89 Eastwood	Mashpee	eh							05/31/79
Clemens	Vanessa	0063	43 Harper	Yarmouth	e			176443	50	12/31/79		05/12/79
Cohen	Dora	0019	38 Lucille	Yarmouth	e							05/07/79
Considine	Gladys	0026	15 Healy	Yarmouth	h		270766			05/31/79	ceceased	05/31/79
Craig	Helen	0133	79 Sidney	Falmouth	eh	medicatio	876847	277433	43	01/31/80		05/08/79
Craig	Alice	0017	504 University	Harwich	e							05/05/79
Crawford	Alida	0103	5 Little Pocket	Yarmouth	e							05/21/79
Curren	Cindy	0142	62 Parkhurst	Falmouth	h		509556					05/03/79
Cutler	Miriam	0032	21 Woodale	Yarmouth	e			415443	0	04/30/79		05/16/79
Davis	Evelyn	0102	201 Bella Vista	Dennis	e							05/20/79
Denny	Charles	0055	79 Patty	Yarmouth	e							05/08/79

FIGURE D-19. MASTER CLIENT FILE

MASTER CLIENT FILE

Last Name	First Name	ID	Address	City	Class	SpecNeed	Med#	XX#	Units	Term Date	Term Why?	EditDate
Denton	Mildred	0136	11 Woodale	Falmouth	e							05/05/79
DiLuzio	Helen	0087	196 Mayflower	Yarmouth	e			537443	08	06/30/79		05/10/79
Dodd	Madeline	0043	143 Delk	Harwich	e	o/u drugs				05/31/79	bill overdue	05/31/79
Donadio	Ethel	0002	57 Hillglen	Sagamore	h		070980					05/04/79
Earl	John	0075	47 Eustic	Dennis	h							05/01/79
Easler	Joachin	0163	84 Fuller	Bourne	eh	medicatio	278530					05/21/79
East	Sarah	0084	12 Dunbarton	Yarmouth	eh		184621					05/07/79
Edelstein	Dorothy	0067	140 Ruth Ann	Yarmouth	e							05/12/79
Evans	Hilda	0025	21 Lockhart	Yarmouth	eh			443443	13	10/30/79		05/09/79
Fischer	Florence	0037	901 Joyce	Dennis	eh	escort						05/20/79
Fleishman	Olive	0094	9 Trailridge	Dennis	e							05/16/79
Fuoco	Andy	0156	1437 Live Oak	Pocasset	weh							05/16/79
Gedman	Corinne	0069	49 Andrea	Hyannis	weh		241280					05/14/79
Glen	Charlotte	0020	25 Camp Drive	Brewster	h							05/08/79
Golden	James	0085	49 Eastwood	Dennis	e	escort						05/08/79
Grieble	Clarke	0080	90 Hermitage	Falmouth	eh	structure	548322			05/25/31	eligibility	05/04/79
Griffin	Geneviev	0141	4 Poppy	Falmouth	e							05/02/79
Griswold	Ethel	0037	507 Anita	Dennis	e		036424	415833	02	05/31/79		05/15/79
Grollman	Ida	0093	65 Bedford	Dennis	e							05/15/79
Guttierez	Isabelle	0012	14 Marigold	Yarmouth	weh							05/31/79
Harman	Robert	0137	46 Wyatt	Falmouth	eh	escort	614760					05/04/79
Hawkins	Eileen	0101	54 Syracuse	Yarmouth	e							05/19/79
Heinz	Richard	0073	1 Ashburton	Hyannis	weh		983379					05/18/79
Helms	Helen	0114	100 Cowing	Yarmouth	e							05/22/79
Hillegass	Agnes	0030	408 Highland	Yarmouth	e							05/14/79
Hoffman	Manuel	0128	149 Firview	Falmouth	eh		258155					05/12/79
Humphrey	Bertha	0108	14 Northview	Dennis	e	medicatio		137453	50	06/30/80		05/27/79
Hunt	Elizabeth	0022	Tom's Trail	Yarmouth	e							05/10/79
Hunt	Mary	0138	71 Belhaven	Falmouth	eh			969433	23	09/30/79		05/03/79
Hurst	Anna	0099	101 Albany	Yarmouth	e			646443	40	03/31/80		05/17/79
Kastens	George	0111	1938 Hurricane	Dennis	eh							05/25/79
Kaufman	Geneviev	0070	34 Chesley	Hyannis	eh	escort	362652					05/15/79
Kerry	Sheila	0074	82 Grosvener	Hyannis	e							05/19/79
Kiley	Mildred	0035	14 Clover	Yarmouth	h	ro phone						05/18/79
Laoon	Peter	0131	1501 Beacon	Falmouth	eh							05/10/79
Lawson	Madeline	0023	50 Valley	Yarmouth	e							05/07/79
Leahy	William	0104	948 Highland	Centervill	eh		926711			05/31/79	moved	05/31/79
Locke	Cathleen	0016	17 Fringewood	Yarmouth	weh							05/04/79
Loiselle	Luente	0078	14 Mark	Dennis	weh							05/04/79
Long	Yolanda	0125	3 Lewis	Mashpee	eh		837683	441043	25	09/30/79		05/14/79
Luger	John	0123	Fishfry	Falmouth	wh		525048					05/15/79

FIGURE D-19. MASTER CLIENT FILE (CON'T)

Last Name	First Name	ID	Address	City	Class	SpecNeed	Med#	XX#	Majits	Term Date	Term Why?	EditDate
Lyons	Joseph	0161	1945 Warm Sprin	Sandwich	en		097738					05/20/79
Marcus	Mary	0117	5 Piper	Centervill	e							05/21/79
Markovsky	Nancy	0107	9 San Felipe	Mashpee	n	1 mxt boo						05/28/79
Mathias	Milton	0135	29 N	Pocasset	wn		851584					05/06/79
Meyer	Ralph	0046	63 westshore	Hyannis	eh	no phone						05/01/79
Mills	Ann	0076	81 Pinecrest	Yarmouth	eh		552398					05/02/79
Mitchell	Rhoda	0113	29 Shaw	Cotuit	en							05/23/79
Mondale	James	0132	3 Live Oak	Buzzards B	wn		236160					05/09/79
Monihan	Helen	0065	49 Fenwick	Brewster	wen	no phone						05/11/79
Morris	Ethel	0026	2 Sweetwater	Dennis	e						deceased	05/12/79
Moses	Roger	0151	14 Lakedale	Hyannis	wn		880612					05/13/79
Mufson	Florence	0083	14 Junius	Hyannis	e	no phone						05/06/79
Mufson	Doris	0058	8 Mill River	Yarmouth	en							05/10/79
Murry	Gla	0166	71 Wasnum	Hyannis	e							05/31/79
Newman	James	0039	9 Flamingo	Yarmouth	e							05/21/79
Nichols	Nicholas	0100	20 Maverick	Yarmouth	e			375643	40	12/31/79		05/18/79
Nunn	Ann	0050	54 Longfellow	Yarmouth	e							05/25/79
Ojeda	Raymond	0126	120 Nashua	Pocasset	e		699410					05/13/79
Ormsby	Rosina	0029	1935 Towneast	Yarmouth	e	structure						05/13/79
Parchuke	Alice	0086	Windmill Villag	Dennis	e							05/09/79
Peeke	Mary	0040	11 Cumberland	Yarmouth	e		120922					05/22/79
Pell	Mary	0007	36 Barbara	Yarmouth	en	escort						07/02/79
Pell	Nell	0011	70 Summer	Yarmouth	e	structure						05/25/79
Percy	Mildred	0052	1 East wharf	Dennis	e							05/14/79
Perlmutter	Ida	0112	19 Whitehaven	Dennis	e							05/24/79
Proxmire	Harry	0047	17 Lancing	Yarmouth	e	no phone						05/02/79
Pryor	Avis	0120	10 Pilgrim	Pocasset	en		742236					05/18/79
Pryor	Eugene	0121	34 Flouncer	Hyannis	eh		235459					05/18/79
Quicne	Melody	0018	5 Babalas	Orleans	n							05/06/79
Rand	Irene	0140	9 St Francis	Falmouth	e							05/01/79
Remy	Marie	0010	19 Manew	Yarmouth	en		270200					05/23/79
Rice	Aaron	0048	4 Wyatt	Harwich	en		541275					05/03/79
Roth	Marion	0159	1602 Pennsylvan	Centervill	weh							05/18/79
Rudman	Robert	0025	137 Main	Chatham	e							05/17/79
Russo	Rov	0033	208 Brodfield	Yarmouth	en							05/17/79
Sandman	Theresa	0147	1507 Vinewood	Hyannis	n	structure	398790					05/09/79
Sarpannes	Richard	0106	46 Englewood	Mashpee	n	1 w/aunt						05/29/79
Sasser	Cresoula	0145	101 Lobster	Falmouth	en							05/07/79
Seal	Marie	0124	70 Moreland	Centervill	en		646338					05/15/79
Sharff	Irene	0036	61 Stallcup	Harwich	e			483533	0	02/28/79		05/19/79
Sherman	Milton	0119	1 Cold Ridge	Bourne	e	medicatio						05/19/79

FIGURE D-19. MASTER CLIENT FILE (CON'T)

MASTER CLIENT FILE

Last Name	First Name	ID	Address	City	Class	SpecNeed	Med#	XX#	Units	Term Date	Term Why?	EditDate
Smith	Albert	0015	31 Mousley	Yarmouth	eh			111338	12	03/31/79		05/03/79
Specter	John	0004	64 Vinton	Hyannis	e							05/14/79
Stafford	Julia	0130	3300 Natura	Falmouth	en			843343	50	04/30/80		05/10/79
Stanley	Harriet	0139	191 Hill Glen	Falmouth	wen							05/02/79
Stapleton	Elizabeth	0105	29 Olson	Dennis	weh			768833	12	06/30/79		05/30/79
Stennis	Helen	0109	23 Wyatt	Yarmouth	e		289736	116443	30	01/31/80		05/26/79
Stewart	Bertha	0089	14 Norman	Harwich	e							05/11/79
Tanner	Ann	0068	7 Monticello	Hyannis	eh	medicatio	411282					05/13/79
Thatcher	Kathryn	0013	48 Emerald	Dennis	en			111347	15	11/30/79		05/01/79
Theobold	Roger	0054	134 Forest	Hyannis	weh							05/07/79
Tower	Nicholas	0134	27 Breaker	Pocasset	weh							05/07/79
Walsh	Mary	0024	26 Sinclair	Dennis	e			647643	09	06/01/79		05/06/79
Warner	Nellie	0144	13 Pearl	Falmouth	en		543987					05/06/79
Weiker	Catherine	0165	21 Cabot	Cotuit	weh							05/31/79
Winkler	John	0127	93 Beth	Falmouth	en			232433	17	09/30/79		05/12/79
Wise	Julia	0098	158 Tahoe	Yarmouth	e							05/15/79
Young	Margaret	0061	48 Sheryl	Yarmouth	en	medicatio				05/31/79	overdue bill	05/11/79
Zukin	Katherine	0051	108 Lawther	Hyannis	en		203986					05/06/79

FIGURE D-19. MASTER CLIENT FILE (CON'T)

APPENDIX E
BLANK FORMS

This Appendix contains blank forms for your use in manually designing your R:base data base. The following forms are provided:

- Attribute Definition
- Report Layout

ATTRIBUTE DEFINITION FORM

Attribute	Name	Type	Length	Key

FIGURE E-1. ATTRIBUTE DEFINITION FORM

FIELD NAME	LINE NUMBER	ATTRIBUTE OR VARIALBE NAME	DATA LINE NOS.	BEGIN NAME COL.	END NAME COL.	NAME WIDTH	BEGIN DATA COL.	END DATA COL.	DATA WIDTH	REPORT COMPUTATION

FIGURE E-2. MANUALLY LAYOUT OUT THE REPORT DESIGN

APPENDIX F
NEW R:BASE FEATURES

F-1. INTRODUCTION

The new R:base version 1.11 is an upgrade of the original version 1.01, incorporating several new interesting features that can be summarized as follows:

- local variables can be used to support conditionals, computations, to move data from one relation to another, and to read input from the keyboard;
- conditional and "loop" processing commands to be used in creating command files for customizing applications and repetitive processes;
- file processing commands to be used in creating command files and looking at them, i.e., DOS commands available directly from R:base;
- a new relational command to add rows from one relation to the end of another relation.

Each feature is explained below.

F-2. LOCAL VARIABLES

Local variables are created by assigning them a value with the "set", "fillin" or "compute" commands. Their values may be reset or recomputed at any time by using the same three commands. The syntax of these commands is:

"SET VARIABLE varname TO [value] (IN relname) (WHERE...)"
varname
expression
attname

"FILLIN varname USING "prompt message" (AT x y)"

"COMPUTE varname AS [COUNT] attname FROM relname (WHERE...)"
MIN
MAX
AVE
SUM
ALL

Where:

- varname is the name given to the local variable (max. 8 characters);
- attname is the name of an attribute;
- relname is the name of a relation;
- "prompt message" is the message used to prompt the operator to type the value of the local variable being defined;
- x, y are the coordinates (x<=24, y<=80) representing the row and column where the prompt message will begin on the screen.

F-3. CONDITIONAL PROCESSING

The "IF" command is used in a command file to conditionally process a series of commands. It tests a set of conditions and, if those conditions are met, then all command lines up to the corresponding (and obligatory) "ENDIF" command are processed once. The syntax is:

```
"IF condition1 ([AND] condition2...) THEN
                OR
.
.
.
ENDIF"
```

The conditions are specified in the same way as in the WHERE clause, except that local variables are tested, rather than attributes in a relation. The conditions are formed by comparing a variable name to a value, another variable name or an expression. The "WHILE" command has the same structure of the "IF" command, substituting "IF" for "WHILE" and "ENDIF" for "ENDWHILE". The difference is that all commands between the "WHILE" and the "ENDWHILE" are processed repeatedly until the conditions in the "WHILE" statement are no longer met.

The "QUIT" command is used to terminate all "WHILE" and "IF" blocks, exiting the command file and returning control to the keyboard. The "QUIT" command then closes all active command files.

F-4. FILE PROCESSING

The "DIR" command works the same as in MS-DOS, i.e., it provides a directory of the files in the specified disk drive. The MS-DOS conventions about wildcards, etc. are applicable here.

The "TYPE" command also works as in MS-DOS, displaying the contents of the specified file on the screen.

The addition of these commands are major improvements. For example, under the old system, it was necessary to exit R:base to check the file directory to make sure file size was not too large to prevent specific operations. This function can now be performed while remaining in R:base.

F-5. NEW RELATIONAL COMMAND

The "APPEND" command takes rows from one relation and appends them to the end of another relation. Unlike the other relational commands, "APPEND" will not create a new relation; instead, the destination relation is permanently enlarged to include the new rows. The syntax is:

```
"APPEND relname1 TO relname2"
```

This command is very useful in adding a day's data entry onto a master relation. For example, instead of "unioning" the various vehicle schedule files to form yet a third file, you could simply combine two files into one. This will greatly lessen the disk storage requirements of the scheduling function.


HE 203 .A56
Commercial
applicatio
Form DOT F 172
FORMERLY FORM D

DOT-I-84-51

TECHNOLOGY SHARING

A Program of the U.S. Department of Transportation